# High Efficiency Prediction Methods for Current and Next Generation Video Coding

Saverio G. Blasi

PhD Thesis

First Supervisor: Prof. Ebroul Izquierdo
Second Supervisor: Dr. Pengwei Hao

School of of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

April 2014

# Statement of Originality

I, Saverio G. Blasi, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third partys copyright or other Intellectual Property Right, or contain any confidential material. I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis. I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university. The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Saverio G. Blasi

01/08/2014

Details of collaboration and publications:

All papers published while working on this thesis are listed at the end of the thesis. Any publication produced in collaboration with others is clearly mentioned.

# Abstract

Consumption and production of video signals drastically changed in recent years. Due to the advances in digital consumer technology and the growing availability of fast and reliable internet connections, an increasing amount of digital video sequences are being produced, stored and shared every day in different parts of the world. Video signals are inherently larger in size than other types of multimedia signals. For this reason in order to allow transmission and storage of such data, more efficient compression technology is needed. In this thesis novel methods for enhancing the efficiency of current and next generation video codecs are investigated. Several aspects of interest to video coding technology are taken into account, from computational complexity and compliance to standardisation efforts, to compression efficiency and quality of the decoded signals. Compression can be achieved exploiting redundancies by computing a prediction of a part of the signal using previously encoded portions of the signal. Novel prediction methods are proposed in this thesis based on analytical or statistical models with the aim of providing a solid theoretical basis to support the algorithmic implementation. It is shown in the thesis that appropriately defined synthetic content can be introduced in the signal to compensate for the lack of certain characteristics in the original content. Some of the methods proposed in this thesis aim to target a broader set of use cases than those typically addressed by conventional video coding methods, such as ultra high definition content or coding under high quality conditions.

# Acknowledgments

First of all I would like to thank my supervisor Prof. Ebroul Izquierdo, for giving me the opportunity of pursuing a PhD in the Multimedia and Vision research group and for his constant guidance and reliable advice during these years. I would also like to thank my second supervisor Dr. Pengwei Hao for his many suggestions and encouragements during these years. I am deeply grateful to Dr. Eduardo Peixoto, who has worked closely with me on some of the topics of this thesis. Without his friendly support I could not have completed this work.

I would like to thank Dr. Marta Mrak for giving me the great opportunity of working with her, and for guiding me while developing some of the contributions described in this thesis. Also my sincerest thanks go to Dr. Matteo Naccari, Andrea Gabriellini and the rest of the Video Coding team at BBC R&D, for their reliable advice and the incredible patience they shown while answering each of my (endless) questions and doubts. I spent some incredible months and learned so much while working there. I really felt part of the team.

Special thanks go to all the friends I was lucky enough to meet at Queen Mary and especially in the MMV lab (too many names!). Pursuing a PhD requires patience and dedication and in many difficult times they helped me find both. I would also like to thank all my other great friends in London, who certainly made my life here better. Finally, I must thank my friends in Rome and those spread in other parts of the world, during these years maybe far from my eyes, but not from my mind.

But most of all, I am grateful to my family.

# Table of Contents

# List of Figures

# List of Abbreviations

| | |
|---|---|
| AMP | Asymmetric Motion Partitions |
| AMVP | Advanced Motion Vector Prediction |
| AVC | H.264/AVC - Advanced Video Coding |
| B | Bipredictional Predicted Inter |
| BD | Bjontegaard Difference |
| CABAC | Context Adaptive Binary Aritmethic Coding |
| CAVLC | Context Adaptive Variable Length Coding |
| CBF | Coded Bit Flag |
| CIF | Common Intermediate Format |
| CTU | Coding Tree Unit |
| CU | Coding Unit |
| DCT | Discrete Cosine Transform |
| DF | Deviation from Flatness |
| DST | Discrete Sine Transform |
| EIP | Enhanced Inter-Prediction |
| FD | Fractional Difference |
| FPS or fps | Frames per Second |
| GOP | Group of Pictures |
| HD | High Definition |
| HEVC | H.265/HEVC - High Efficiency Video Coding |
| HM | HEVC test Model |

| | |
|---|---|
| I | Intra |
| IDR | Instantaneous Decoder Refresh |
| ITU | International Telecommunication Union |
| ITU-T | International Telecommunication Union - Telecommunication Sector |
| JCT-VC | Joint Collaborative Team on Video Coding |
| JM | Joint AVC test Model |
| JPEG | Joint Photographic Experts Group |
| JVT | Joint Video Team |
| LMS | Least Mean Squares |
| MPEG | Moving Picture Experts Group |
| MSE | Mean Square Error |
| MV | Motion Vector |
| MVD | Motion Vector Difference |
| PSNR | Peak Signal to Noise Ratio |
| P | Unidirectional Predicted Inter |
| PU | Prediction Unit |
| QP | Quantisation Parameter |
| RAP | Random Access Point |
| RD | Rate-Distortion |
| DPCM | Differential Pulse Code Modulation |
| RQT | Recursive Quad-Tree |
| SAD | Sum of Absolute Differences |
| SAO | Sampled Adaptive Offset |
| SC | Sampled Curvedness |
| SSIM | Structural Similarity Index |
| TU | Transform Unit |
| UHD | Ultra High-Definition |
| WP | Weighted Prediction |

***Question**: "How many seconds of uncompressed full HD video can I fit on a DVD?"*

***Answer**: "Due to colour subsampling, a single uncompressed frame in full HD resolution is* $(1920 \times 1080) \times 16$ *bits* $= 4.2$ *MB. A DVD is* $4.7$ *GB, so you can fit about* $238$ *frames. At* $60$ *frames per second, this is around* $4$ *seconds of video."*

from "Yahoo Answers", 2010

# Chapter 1

# Introduction

In the last decades production and consumption of multimedia information have drastically changed. Only a few years ago the general public was accessing video content mainly through analogue means such as the traditional television signal, movie picture films or videotape-based cassettes. Today, we are surrounded by devices and services capable of generating or displaying digital videos: computers, tablets and smartphones, cable or satellite TV box-sets, DVD and Blu-Ray players, videogame consoles, digital video distribution services, video-conferencing, camcorders, cinemas, even billboards in bus stations. In developed markets the average consumer is today able to capture a high-quality video with a mobile phone and share it instantly on the internet, all of this maybe using a portable device with a mobile connection.

Several factors played a role in this process. First, the consumer technology market quickly evolved to the point where cheap, small and yet powerful electronic devices are today widely available, capable of producing and consuming digital multimedia content. Second, a larger than ever percentage of the population has access to a fast and reliable internet connection: recently some countries even started offering ultra-fast mobile data connections such as 4G LTE. Finally and most importantly, the incredible advances in the field of multimedia signal processing provided the bases to allow for storage and

distribution of multimedia content at rates and qualities that were unimaginable only ten years ago.

One of the most critical aspects that differentiates digital video sequences from other types of multimedia signals is that they are relatively very big. For example, an uncompressed 4-second sequence in high definition can easily take up more than 4.5 gigabytes of storage space. At this rate a two hours movie would require several terabytes. Accessing or transmitting such raw data is clearly not viable, and for this reason processing of video signals is necessary especially with the goal of obtaining compression. Most applications that involve the consumption of video signals must deal with compressed data: this is not limited to consumer-oriented services such as mass content distribution via digital TV or satellite, or physical storage like DVDs or Blu-Rays. It is also essential for a variety of professional applications such as the transmission of signals throughout the video production chain, medical imaging, video surveillance, and so on. Due to the growing popularity of all these applications, a lot of research over the past twenty years has focused on satisfying the needs for more efficient processing of digital video signals.

The branch of signal processing that deals with these problems is usually referred to as video coding. Notice that this is by no means a simple task for several reasons, only some of which are listed here:

1. Especially in the case of consumer applications, the typical user has limited hardware resources, both in terms of storage space and computational power. Consider for instance the example of a digital video camera: the device should be capable of compressing the captured footage in real time in order to use as little storage as possible, while still preserving the best possible quality of the signal. This, while consuming as little battery power as possible.

2. Video signals are very dissimilar one from the other. Their characteristics strongly depend on the way they are generated. For instance, sequences containing only natural scenes captured with a recording device have very different characteristics

than signals containing screen content (e.g. subtitles or other computer-generated elements) or completely artificial signals. Video coding must be able to adapt to such differences and achieve reasonable efficiency under a broad variety of conditions.

3. The market is going faster than expected, and video coding technology must keep up with it. For instance full high-definition was considered cutting edge only a few years ago, and yet devices are already available capable of producing and displaying content at even higher definitions. A growing number of users will soon start demanding for distribution of content at such definitions. Responding to such expectations can be extremely difficult for content providers, which must deal with an exploding demand while still relying on the same old physical infrastructures.

4. As opposite to other fields in which there is an explicit separation between pure research and its application, the fast pace at which new technology is needed by the related industry makes such separation much narrower in video coding. This is mostly due to the strong impact of video coding standardisation. Video coding standards have been and are being developed to allow greater inter-operability among different devices and services. It is thanks to these efforts that a great deal of innovation has been made available to the general public in a relatively short amount of time. Meaningful research in the field should take into consideration some of these aspects, possibly targeting integration of new technologies within current or next generation standards.

The work presented in this thesis aims at introducing novel technologies, developed with the general goal of providing high efficiency video coding while also taking into account all the aforementioned issues. The thesis mostly focuses on improvements to the prediction module used to exploit spatial and temporal redundancies, due to the fact that in typical schemes his is the component responsible for providing the highest amount of compression efficiency as will be extensively detailed in the rest of this thesis. Several aspects of video coding are targeted in the thesis, such as complexity of encoding

schemes, compression efficiency, or video compression under specific conditions. Most of the proposed schemes are derived based on theoretical or statistical models, with the aim of providing a theoretical basis to support the various practical implementations. It is shown in the thesis that conventional prediction schemes used in state-of-the-art video coding suffer from several issues which limit their effectiveness depending on the application. Many methods are proposed with the purpose of improving or replacing such schemes, with the goal of providing efficient compression while preserving as much as possible the quality of the compressed data. Some of these contributions involve the definition of appropriately defined synthetic content, and it is proved in the thesis that in some cases external synthetic content can be injected into the signal to compensate for the lack of certain characteristics in the original content. Also, video coding at very high levels of quality is considered, to address the growing needs of specific professional and consumer applications.

The main motivation for carrying out such a work is simple. The efficiency of video coding technology is growing exponentially at a pace that was completely unanticipated. The effects of such technological breakthroughs can be observed in the world today in a factual, visible way. All this happened only thanks to the hard work and combined efforts of many researchers working in the field.

The rest of this thesis is organised as follows:

**Chapter 2** presents a background on video coding technology, with particular emphasis on state-of-the-art prediction methods. Also, a brief overview of the main standards used to study, implement and test most of the techniques proposed in this thesis is presented in the chapter.

**Chapter 3** illustrates the proposed techniques for efficient low-complexity inter-prediction via adaptive precision sub-pixel motion estimation, based on geometrical characterisation of the residual error surface and binary classification.

**Chapter 4** presents the proposed enhanced inter-prediction module via parametric trans-

formations of the prediction candidates. Several methods are presented for implementing the module at different stages in the coding scheme. The method is integrated in the context of widely available current and next generation video coding standards.

**Chapter 5** illustrates the proposed techniques for high quality video coding via transform domain prediction methods. An analysis of conventional prediction techniques is presented, showing that conventional methods may not be sufficiently efficient under high quality constraints. Different techniques are proposed and presented in the chapter to address these issues.

**Chapter 6** presents some general conclusions and observations about this work. Some future developments for improving and expanding the proposed contributions are also introduced in the chapter.

# Chapter 2

# Background

In this Chapter, some general video coding concepts are first presented. A review of state-of-the-art prediction methods is also included in the chapter. Finally, an overview of the H.264/AVC and H.265/HEVC standards is presented in the last part of the chapter with particular emphasis on the prediction modules included within these standards.

## 2.1 Video Coding Concepts

### 2.1.1 The Encoder-Decoder Framework

Video coding is the application of source coding to digital video signals. Source coding (or data compression [1]) consists of describing a signal using less bits than its original representation while still preserving the ability of consuming it. Consequently video coding can be defined by the two mutual processes of transforming (compressing) an input video signal into a smaller amount of data (the bitstream), and then transforming it back (uncompressing it) to a reconstructed sequence ready for consumption. The main goal of video coding technology is that of increasing as much as possible the compression ratio between the size of the original data and the size of the bitstream.

Two types of compression can be considered depending on the nature of the uncompressed signal with respect to the original one. In case these two signals are identical, namely it is possible to decode the bitstream into an exact replica of the original signal, the transformation is referred to as lossless compression. Conversely the transformation is referred to as lossy compression in case the uncompressed signal is not identical to the original one. Lossy compression algorithms are typically capable of providing higher compression ratios than lossless algorithms. Therefore due to the fact that video signals may be very large compared with other types of media signals, lossy compression is widely used in video coding.

The entity that takes care of transforming the original signal into a bitstream is usually referred to as the encoder [2]. Modern video encoders make use of extremely complex algorithms to define and optimise each step in this transformation, and yet most of these algorithms are based on a single concept [3]: compression can be obtained removing "unnecessary" portions of the signal based on exploiting well-known phenomena, such as those illustrated in the following examples.

1. The human visual system is less sensitive to perceive variations in colour than in brightness. Such lower acuity for colour differences may be exploited to achieve compression, for instance by using less data to represent the colour information (at the cost of higher losses) than to represent the brightness. Compression is achieved by removing irrelevant portions of the data. Note that this phenomenon and how it is exploited in conventional coding schemes will be further detailed in the following of this chapter.

2. Consider the case of a black and white (binary) image, as the one shown on the left of Figure 2.1. Uncompressed binary images are typically represented as a succession of bits $b(i, j)$ to represent the appearance of a given location in the image. Consider then a transformation that arrange such bits in a row vector and returns the integers $N_0, N_1$ as the number of successive identical bit values. Applying this transformation on the image in Figure 2.1 would result in a smaller

representation (the two integers $N_0, N_1$) which fully describe the content of the original signal. Compression is achieved removing redundant portions of the data.

Most video compression schemes are based on exploiting these phenomena. Depending on the complexity of the related methods, the resulting bitstream can be drastically different than the original signal. Such data needs to be transformed back to a representation ready for consumption. The entity responsible for this transformation is referred to as the decoder. The term codec is often used to refer collectively to the combination of an encoder and a suitable decoder, as illustrated in Figure 2.2.

The encoder-decoder framework is of fundamental importance for video coding especially in case the compressed signals need to be shared, for instance in content distribution or storage in physical media. Clearly the users that receive the compressed signal should be able to decode it. For this reason, and in order to avoid the coexistence of an endless variety of compression schemes each requiring a different decoder, many efforts in developing video coding technology converge in the definition and ratification of video coding standards. A video coding standard typically consists of a set of specifications

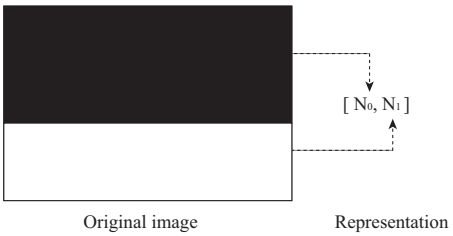



Figure 2.1: A black and white image (left) and its representation (right).

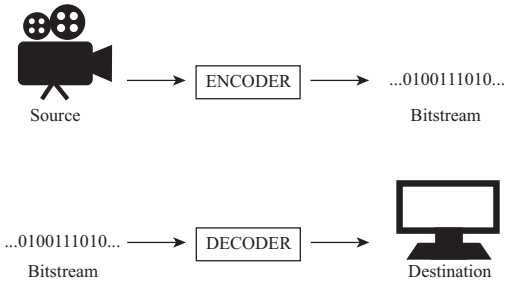



Figure 2.2: The codec framework.

(or normative requirements) which regulate the steps necessary to decode a bitstream into a reconstructed sequence ready for consumption. Most standards are developed by experts in the field working together with the goal of providing the best technology to achieve compression under a wide range of conditions and to address the relevant industry needs. In most cases only the decoder (and the corresponding bitstream) is defined in the specifications. Each manufacturer or service provider can independently develop a suitable encoder as long as it produces compliant bitstreams. The standardisation process is essential to ensure that the best compression technologies can be quickly and widely adopted by the industry.

### 2.1.2 Colour Spaces and Video Formats

The perceived quality of a video sequence is strongly affected by the characteristics of the corresponding digital signal. Uncompressed digital image and video signals are usually represented as a succession of picture elements (pixels); each pixel is a digital representation of the physical appearance of the data at a specific location. When dealing with grayscale content, a single binary string can be used for each pixel to represent the grey level (or intensity). Each intensity is mapped to a string of a fixed length (the bitdepth). Bitdepths of 8 or 10 bits with correspondingly 256 or 1024 intensity levels are commonly used in video and image processing.

When considering colour content, the colour information must also be mapped to a digital representation. When such abstract maps are extended to include the chromaticities of the colour components (namely an objective description of their appearance), they are referred to as colour spaces. For instance several colour spaces can be defined based on the RGB colour map (in which the colour information is mapped to a combination of red, green, and blue intensity levels). The sRGB colour space is one of the most widely adopted [4].

The human visual system is said to be more sensitive to brightness variations (referred

to as luminance, or luma) than it is to colour (referred to as chrominance, or chroma) [5]. Unfortunately the RGB colour map (and the related colour spaces) represents luma and chroma jointly in each component. For the purposes of compression it may be reasonable to exploit such difference in sensitivity and process luma and chroma separately. For this reason the YUV colour map has been defined. Note that the term YUV is not an abbreviation but comes from historical reasons, and its use is sometimes questioned [**?** ]. A pixel in YUV is also represented by three components as is the case for RGB. The Y component describes the luma, achromatic characteristics of the image; the U and V components represent the chroma information. YUV is widely used in video coding, and for this reason it was always used in this thesis unless otherwise stated. A pixel in YUV can be easily converted to RGB by means of simple formulas.

A technique is often used to take advantage of the YUV colour map referred to as chroma subsampling [6]. The chroma components of the video signal are subsampled with respect to the corresponding luma. Several schemes have been defined and standardised at this purpose: in the $4:4:4$ format no chroma subsampling is used; in the $4:2:2$ format the U and V components have half the size of luma in the horizontal direction, and full size in the vertical direction; in the $4:2:0$ format one chroma value is considered for each $2 \times 2$ luma. Note that the $4:2:0$ scheme is commonly used formats in video coding applications. When using a bitdepth of 8 bits, in average only 12 bits per pixel are necessary instead of 24 bits needed with the $4:4:4$ format.

Digital video sequences are discrete signals formed of a countable succession of digital images of a fixed size (each image is referred to as a frame), rapidly displayed one after the other. Two methods can be typically used to display a sequence. In interlaced scan only half of the pixels in a frame (alternatively the even or odd rows) are displayed at a given time instant; the array containing the pixels displayed at each time instant is referred to as a field. Conversely, the frame is shown entirely at each time instant when using progressive scan. The use of interlaced scan is rapidly decreasing and it is of little interest for the methods proposed in this thesis, therefore progressive scan is

assumed everywhere in this thesis unless otherwise specified. The rate at which frames are displayed (framerate) is crucial to the perceived visual quality. Some studies appeared in the past with the goal of proving that the human visual system can perceive separately no more than a small limited number of images per second displayed successively [7]. For this reason until very recently video signals have been mostly captured and broadcast at framerates of 25 to 30 frames per second (fps); such rates have been used as an almost universal industry standard since the first half of the 20-th century. In practice the human brain processes visual information in a way that is not yet completely understood. In recent years, advances in neurosciences allowed for a deeper understanding of the way human brain processes rapid changes of visual information [8] [9], indicating that while displaying more frames per second not necessarily corresponds to a more pleasant experience for the viewer, higher frame rates than 30 fps can be beneficial to the visual quality. For these reasons the market is adopting higher frame rates: the film industry is experimenting with rates of 48 fps, while recent recommendations for broadcasting standards include extremely high framerates, up to 120 fps [10].

The resolution of a video sequence (the number of pixels in each frame) is another factor to impact the perceived quality. Video resolutions are typically standardised in video formats [11]; conventionally the resolution only specifies the pixels in the luma component (where chroma is derived according to the subsampling). An important family of video formats is the Common Intermediate Format (CIF), including CIF ($352 \times 288$ pixels), QCIF ($176 \times 144$ pixels) or 4CIF ($704 \times 576$), all with a $4:3$ aspect ratio between width and height of the frames. Also, important formats have been standardised by the Digital Video Broadcasting (DVB), including 720p ($1280 \times 720$ pixels), and 1080p ($1920 \times 1080$ pixels), based on a wider $16:9$ aspect ratio [12]. Recently, larger formats have been defined typically referred to as ultra high-definition (UHD) formats [10], including the $4K$ UHD format ($3840 \times 2160$ pixels), and the $8K$ UHD format ($7680 \times 4320$ pixels). Video compression plays a very important role in providing sufficient compression ratios to allow distribution of such high resolution content.

### 2.1.3 Distortion Metrics and Video Quality Evaluation

When using lossy compression the reconstructed signal output by the decoder is different than the signal prior to compression. Such difference is usually considered as an undesired distortion added to the original data as a by-product of compression (these by-products are typically referred to as compression artefacts). Typical compression algorithms obtain higher compression ratios at the cost of larger distortion. One of the main goals of video coding algorithms is therefore that of achieving the highest possible compressions while at the same time limiting the distortion in the reconstructed signal (as illustrated in the rest of this chapter).

A crucial problem when targeting this goal is that distortion is very difficult to measure. In fact, distortion is even difficult to define, because it depends on the quality of a signal as it is perceived subjectively. Several distortion metrics have been defined to address this issue. The purpose of a distortion metric is to provide a numerical score to evaluate the quality of the reconstructed signal with respect to the original. Successful metrics used at this purpose are the sum of absolute differences (SAD), sum of squared differences (SSD) or mean square error (MSE). All these methods have been extensively used in signal processing (and video coding) applications due to their simplicity, low complexity and good performances. For instance assuming an original signal $x(i)$ and a reconstructed signal $x_{rec}(i)$ where $i = 0, ..., L$, the MSE between the two signals is defined as:

$$MSE\left(x(i), x_{rec}(i)\right) = \sum_{0}^{L} \left[x(i) - x_{rec}(i)\right]^2 .$$

MSE only requires in average one multiplication and two additions per sample, and it has nice mathematical properties such as convexity and differentiability. In practice MSE is often converted to a logarithmic scale to obtain the well known Peak Signal to

Noise Ratio (PSNR). PSNR is measured in $dB$s, and is formally defined as:

$$PSNR_{dB} = 20 \log_{10} \left( \frac{x_{max}}{MSE} \right),$$

where $x_{max}$ is the maximum allowed value for $x(i)$, referred to as the dynamic range. Higher PSNR values correspond to lower MSE and are therefore associated with better quality of the distorted signal compared with the original signal. PSNR is useful when comparing signals with different dynamic ranges, but otherwise is completely equivalent to the MSE.

Despite their attractive features, the use of SAD, MSE or PSNR has been often questioned due to some fallacies of such methods in measuring quality as perceived by human subjects. A famous example of such fallacies consists in shifting all pixels in an image by one position in any direction: MSE between original and shifted pictures is relatively high, even if the two images are virtually identical to a human subject [13]. For these reasons algorithms have been proposed in the past to define alternative distortion metrics. A well known class of metrics has been proposed based on the assumption that human perception is less sensitive to a particular class of artefacts, namely those that may be caused by sensor fallacies (for instance when capturing the signal with a camera). Examples of such fallacies are brightness variations, spatial shifts or contrast changes. Human eyes are also sensors, therefore the brain should be trained to automatically compensate for these artefacts. Conversely "less natural" artefacts (referred to as structural distortions) such as additive noise, blur, blocking artefacts and so on, can be expected to have a stronger impact on the subjective quality of the signal. Distortion metrics should appropriately weight distortions to take this behaviour into account. An index was defined [14] accordingly, referred to as the Structural Similarity index (SSIM). The multiscale SSIM metric was later introduced [15] as an extension of SSIM capable of adapting to different resolutions of the signals. Also another class of video quality assessment metrics has been proposed referred to as MOtion-based Video Integrity Evaluation index (MOVIE) [16], based on the assumption that motion plays an important

role in quality assessment.

It is worth noting that PSNR and MSE are still the metrics of choice in most of the research in video coding. It is difficult to find references in the literature which do not include results in terms of either of these metrics. This is mostly due to the fact that despite their issues, these metrics are particularly well suited to measure video quality in most of the scenarios of interest for video coding research. A study was presented [17] which highlights the fact that MSE performs very well under a "same content, same codec" scenario, namely when it is used to compare signals encoding the same content and using the same codec, to validate the performance of different tools.

### 2.1.4 Rate-Distortion Theory

Modern video coding schemes perform the transformation from the original signal to the bitstream by selecting an optimal set of coding tools out of a variety of possible options. The best tools for the particular content being coded are chosen and used to produce a bitstream as small as possible. The size of a bitstream is usually measured in terms of the average number of bits necessary to code a second of video (in bits per second), and is usually referred to as the bitrate. As mentioned before most schemes achieve higher compression ratios (i.e. lower bitrates) at the cost of larger distortions. An efficient encoder should be capable of selecting and tuning its tools based on a trade-off between rate and distortion. The discipline that studies how this trade-off can be achieved is referred to as rate-distortion (RD) theory.

Most of the techniques in RD theory are based on Lagrangian optimisation [18]. In particular assume that the encoder is currently selecting which tool, out of a set of possible tools $T_k$ $k = 0, ..., N$, should be used to encode the currently considered portion of the signal. Each tool takes the input signal $x(i)$ and provides a certain bitstream $b_k$ which, once decoded, results in a reconstructed signal $x_{rec,k}(i)$. Define as $D\left(x(i), x_{rec,k}(i)\right)$ a certain distortion metric, as for instance those described in subsection

2.1.3. The Lagrangian cost associated with tool $T_k$ is defined as:

$$J\left(T_k\right) = D\left(x(i), x_{rec,k}(i)\right) + \lambda b_k \tag{2.1}$$

where $\lambda$ is the Lagrangian multiplier. Then the optimal tool $T^o$ can be defined in terms of minimisation of the aforementioned cost, or:

$$T^o : \min_{T_0,\ldots,T_N} \{J\left(T_k\right)\}. \tag{2.2}$$

Most modern video coding schemes make use of Lagrangian optimisation based on Eq. 2.1 and 2.2 to improve the efficiency of the encoding.

Note that typical video encoders allow the user to select an expected target level of quality of the reconstructed signal. In order to address a broad range of applications, the same video codec can be used to target a signal with very low distortion (but consequently high bitrates) or a small signal (at the cost of higher compression losses). In order to evaluate the performance of video compression under different conditions, a wide range of qualities is usually tested. Distortion of the reconstructed signal and rate of the compressed bitstream are measured for each level of quality. Results of such tests can be easily represented by means of so called RD curves, namely plots in which distortion values are represented against bitrates for each tested value of the target quality. RD curves can be used to compare the efficiency of different coding schemes (for instance a proposed method against a state-of-the-art anchor). An example of such a comparison is shown in Figure 2.3 for the PSNR distortion metric. A scheme is more efficient than the benchmark if it provides a curve above the anchor.

Recently, a measure has been proposed to analytically assess this comparison, referred to as Bjontegaard BD-rate [19]. The BD-rate is a measure of difference between the areas

below the RD curves obtained by the two schemes. The larger is this difference, the better is the method with respect to the benchmark in an RD sense.

## 2.2 Block-based Video Coding

### 2.2.1 The Hybrid Model

First attempts at video coding standardisation were developed already in the early 90s mostly to target video-conferencing applications. In these schemes quality of the reconstructed video was often neglected in favour of smaller bitrates. Nowadays, as a response to the needs of the broadcasting and cinema industry, the main goal of video coding is rapidly shifting towards achieving higher qualities of the decoded signals. Regardless of such a complete change of focus and in spite of a radically different technological landscape, most of the video coding standards available today are still based on the same model as those early attempts. Such model, who has proven to be so successful to stand the test of time for so many years, is typically referred to as the block-based hybrid model.

When using this model the original signal is partitioned in blocks of a certain size which are independently encoded. The idea of partitioning an image in blocks prior

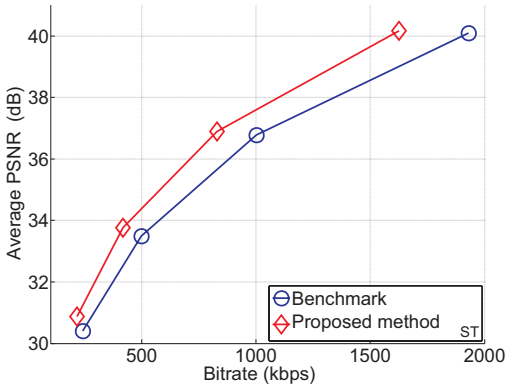



Figure 2.3: RD curve.

to compression directly results from well known concepts of information theory [20]. Assuming some correlation among spatially neighbouring pixels, lower bitrates should be obtained by grouping pixels together as opposite to coding each pixel independently. The size of the blocks is a crucial factor for compression efficiency. Increasing the size of the blocks means that there are fewer blocks in each frame, and the related parameters are coded in the bitstream less frequently. Conversely the correlation among pixels within a block tends to decrease with increasing the block size resulting in possibly higher bitrates.

Each block is successively input to a sequence of three units which sequentially perform the steps needed to transform the original signal into the final bitstream. First, the prediction unit is based on the idea that the content of the current block can be predicted from other parts of the signal due to the presence of redundancy. For instance neighbouring pixels in a frame may have some degree of similarity, especially in areas of largely stationary content. Redundancy among pixels in the same frame is referred to as spatial redundancy, and it is exploited in video coding by means of so called intra-prediction. On the other hand video signals are formed of a succession of frames and it is reasonable to expect some similarity among temporally neighbouring frames. Redundancy among content of different frames is referred to as temporal redundancy and it is exploited by means of so called inter-prediction. Typical video coding schemes make use of intra or inter-prediction methods to provide the best prediction for the current block. This is subtracted to the original block to obtain the residual samples; such residuals can be expected to have lower energy than the original signal and therefore are better suited for compression. Residual samples are input to the transform unit which attempts at finding a more compact representation of the data and then removes selected parts of the signal by means of quantisation. Finally the quantised coefficients are input to the entropy coder, to be transformed into the actual bitstream by means of appropriate binary codes and entropy coding methods. The units in the hybrid model are represented in Figure 2.4 and will be briefly detailed in the rest of this chapter.

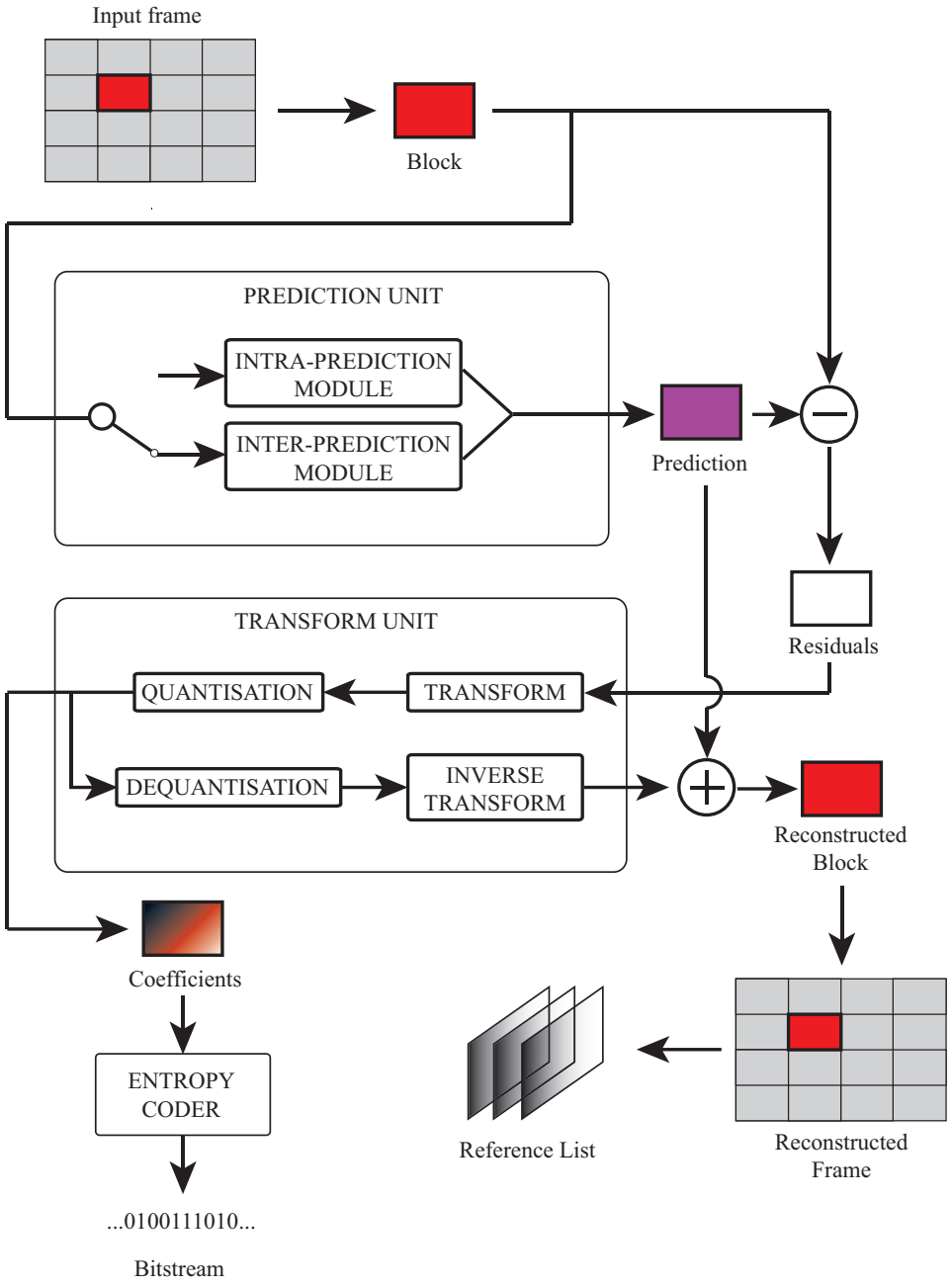



Figure 2.4: Block-based hybrid model.

The information needed to predict the content in the current block is also encoded in the bitstream. At the decoder side, this is extracted and used to compute the prediction, which is added to the decoded residual samples to obtain the reconstructed block. Note that only the content of already reconstructed blocks is available to the decoder to compute the prediction, as illustrated at the bottom of Figure 2.5. To avoid error

propagation while decoding, it is critical that the output of the prediction unit is exactly the same at encoder and decoder side. For this reason, while in theory the encoder could use the original signal to compute the prediction (as at the top of Figure 2.5), in practice the encoder typically keeps track of the decoded signal in the same way as it would be reconstructed by a suitable decoder and uses this signal to compute the prediction.

A brief description of the main components of a typical video encoder is presented in the following subsections.

### 2.2.2 Intra-prediction

Intra-prediction, sometimes referred to as predictive image coding, consists in computing a prediction for the current block using a number of pixels (referred to as reference samples) extracted from the same frame [21]. Redundancy appears most likely among close neighbouring pixels, and for this reason only pixels in the surrounding of the currently encoded block are used as reference samples, as those shown in dark gray in Figure 2.6 (a).

Several methods have been proposed to efficiently compute intra-prediction given the reference samples. Due to the fact that a single frame in the sequence is involved in the process, many of these methods are shared among video and still image coding schemes.

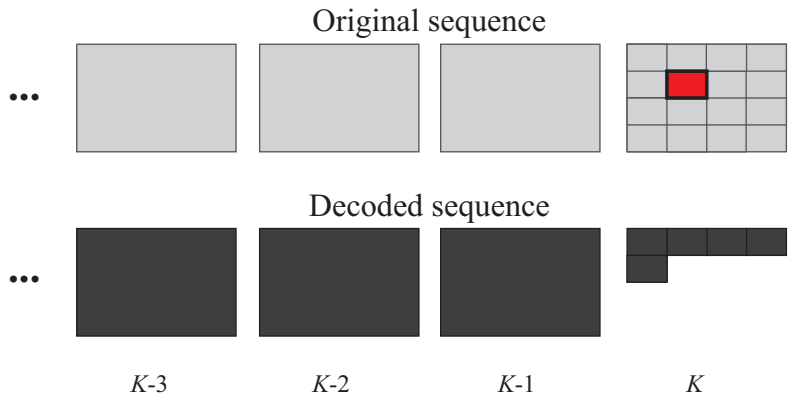



Figure 2.5: Data used at encoder side to compute the prediction.

Most intra-prediction schemes take into account the fact that the signal is successively input to the transform unit to obtain a representation in the transform domain. Given the nature of the transformed signals and the transform methods commonly used, typically the largest coefficients can be expected at low frequencies and particularly at the zero-frequency component, usually referred to as DC coefficient. In order to limit the impact of such large coefficients in the related bitrates, first attempts at intra-prediction mainly targeted prediction of the DC coefficient. Already in the JPEG image coding standard [22] a constant value (equal to half the dynamic range of each pixel) is subtracted from all samples before a block is input to the transform unit. As an extension of this idea DC intra-prediction was proposed and included in later image and video coding schemes. DC prediction simply consists of predicting all pixels in the current block using a single value, typically obtained as the average of all reference samples, as illustrated in Figure 2.6 (b).

A very successful class of intra-prediction methods [74] is based on the idea that objects in an image often follow a direction of propagation. By identifying this direction, reference samples can be projected inside the block to return a good estimate of the actual block content. A simple application of this concept consists in considering only the horizontal and vertical directions. Reference samples located immediately at the left or immediately on top of the current block are copied in the horizontal or vertical direction respectively to form the corresponding prediction. Horizontal prediction is illustrated in Figure 2.6 (c). Extending this idea, reference samples can be interpolated and then projected, to obtain a so called angular prediction for a variety of possible directions. Many modern video coding standards make use of angular intra-prediction with a large number of possible directions, as illustrated later in this chapter.

Intra-prediction algorithms are typically tested in an RD sense following a similar scheme as the one illustrated in subsection 2.1.4. Increasing the number of tested algorithms increases the chances of obtaining an accurate prediction but comes at the cost of higher computational complexity. For instance, allowing 8 angular prediction directions

along with DC prediction results in almost 5 times higher computational complexity than using only DC prediction in state-of-the-art encoders [23]. For this reason methods have been proposed to reduce the complexity of intra-prediction. A technique was proposed [24] based on early termination of angular prediction by discarding directions that are unlikely to perform well. Similarly, adaptive single-multiple intra-prediction [23] was also proposed, based on the idea that if the reference samples are highly correlated one to each other, angular intra-prediction would provide almost same results as DC prediction regardless of the direction, and therefore only DC prediction is tested. Finally, the intensity gradient technique was also introduced [25] to reduce intra-prediction complexity based on a preprocessing of the current block aimed at identifying predominant

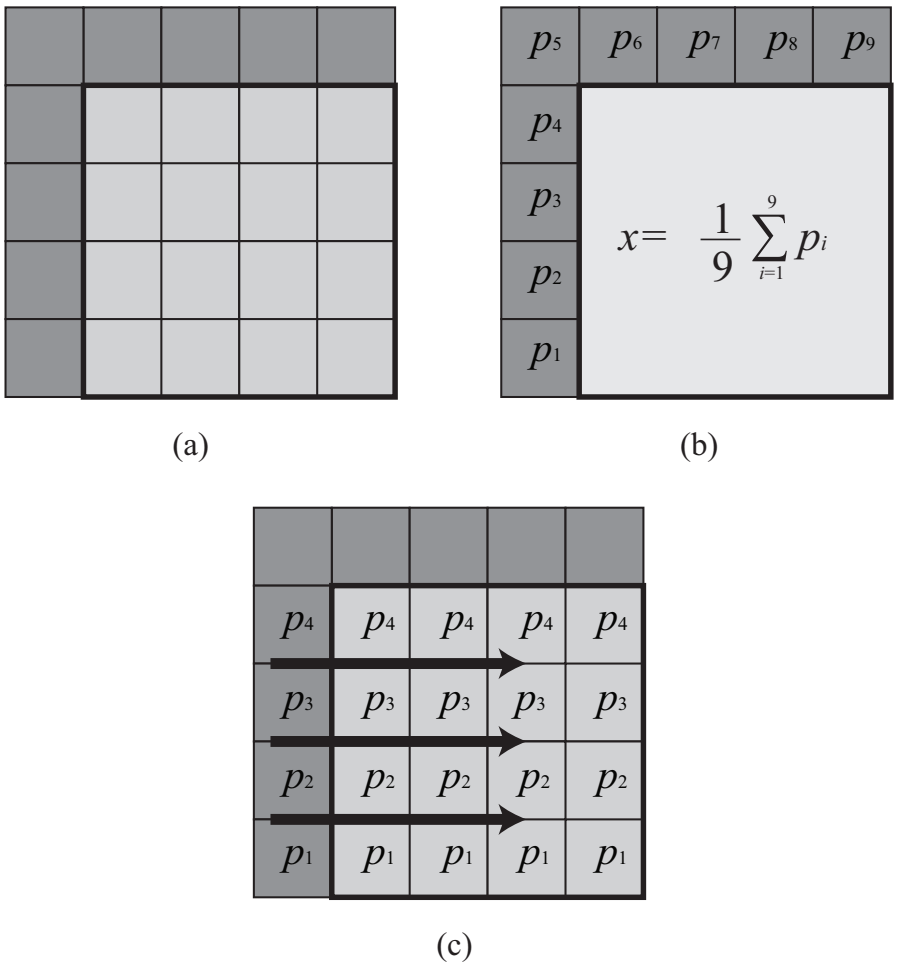



Figure 2.6: Intra-prediction.

directions of propagation of the content. Only angular directions close to the predominant direction are tested.

### 2.2.3 Inter-prediction

Inter-prediction is typically achieved by means of two separate modules referred to as motion estimation and motion compensation respectively [26]. Motion estimation refers to the process of computing the motion information, namely the information necessary to extract a prediction from a previously encoded frame (referred to as reference frame); motion compensation refers to the process of using the data in the motion information to actually compute the prediction. Motion estimation is typically only performed at the encoder side, and the resulting motion information is encoded in the bitstream. The decoder extracts this data and consequently performs motion compensation to compute the prediction for a given block.

As opposite to spatial redundancy which is in general low and located mostly in the close surroundings of the current block, temporal redundancy may propagate throughout a high number of frames and span across a multitude of pixels. Therefore the process of exploiting this redundancy in the best possible way is crucial to achieve the highest compression ratios. At this purpose many video coding standards perform motion estimation by means of a class of techniques referred to as block matching algorithms.

When using block matching algorithms, a prediction of the same size as the current block is extracted from a previously encoded reference frame, by means of a two dimensional array referred to as motion vector. A motion vector identifies the horizontal and vertical displacement between the location of the current block and the prediction block in the reference frame. While performing motion estimation, the encoder considers a set of possible motion vectors. A cost is computed for each motion vector in the set. This may be computed purely in terms of distortion between original and prediction blocks (typically using SAD or SSD) or in terms of an RD metric. The motion vector that

produces minimum cost is selected for motion compensating the current block. Block matching is illustrated in Figure 2.7.

The simplest implementation of this kind of algorithms consists in searching for all possible prediction candidates in the reference frame, considering any motion vector that points to a block within the boundaries of the frame. Such algorithm is usually referred to as exhaustive search (or full search) motion estimation. Exhaustive search algorithms require the encoder to compute a distortion for a very high number of prediction candidates. Especially in case of high resolution sequences, this might not be acceptable in terms of computational complexity and coding time. For this reason practical implementations typically allow only a restricted subset of motion vectors to be tested for each block. A simple way to obtain such restrictions consists in considering a search window, namely limiting motion vector components to a maximum and minimum value. Many other methods have been proposed to reduce the complexity of block matching algorithms, a selection of which is presented in the following of this subsection.

Under the assumption that distortion is computed using SAD, the partial sum of absolute differences method was presented [27] based on the simple idea that the SAD computation can stop as soon as the current partial distortion is higher than the current minimum distortion. More importantly, the successive elimination algorithm was proposed [28] which allows for early termination of the motion estimation search based on simple mathematical manipulation of the current prediction candidate and original block. An extension of the successive elimination algorithm to other distortion metrics

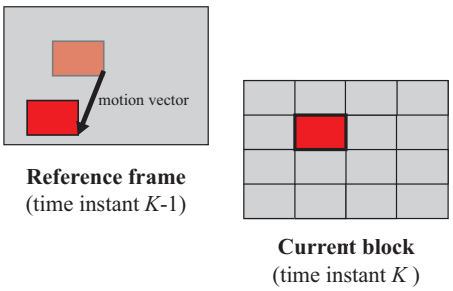



Figure 2.7: Block matching motion estimation.

was later proposed [29], and the method was also applied to other search algorithms than full search [30] [31]. Similarly the block sum pyramid algorithm was proposed [32] also capable of achieving the global optimal solution at faster rates than full search.

These algorithms still return the same solution as full search algorithms. Other methods have been proposed with the different goal of reducing complexity even further at the cost of returning sub-optimal solutions. Some of these methods are based on the assumption that spatially neighbouring blocks may be correlated with the current block. For instance a technique was proposed where the median of motion vectors already selected as optimal solutions in surrounding blocks is used as starting point for a block matching search using a small window [33].

Following again from the observation that temporal redundancy may be spread to span pixels in distant locations from the coordinates of the current block, a class of fast algorithms has been proposed based on the idea of testing a small number of motion vectors pointing to blocks in locations far from the location of the current block, and far away one from the other. Such algorithms rest on the assumption that the distortion between current block and reference frame is approximately a convex function and therefore solutions can be successively refined to obtain a minimum. While the validity of this assumption has been questioned [34], these algorithms have been proved extremely successful and are often used in common video coding schemes. Among these is the three step search [35] illustrated in Figure 2.8. When using this algorithm only locations marked with a square in the figure are initially tested, to find the motion vector at minimum distortion (marked with a black square) among such candidates. Starting from this solution another set of motion vectors is tested (marked with a triangle in the figure) pointing to locations in its surrounding, to find a new motion vector at minimum distortion (marked with a black triangle). This is repeated to test a new set of motion vectors (marked with a circle in the figure) until the motion vector at minimum distortion is returned as final solution (marked with a black circle).

Similarly the diamond search [36] has been proposed, based on two search patterns

(large and a small search pattern respectively) with a diamond-like shape obtained rotating a square by 45 degrees. The large search pattern is initially used to test candidates as in the first step in the three step search. This initial search stops when the solution at minimum distortion results at the location in the centre of the pattern. The small search pattern is then used to refine the solution. The small and large search patterns used in diamond search are shown in Figure 2.9 (a) and (b) respectively. Hexagon search was later proposed [37], based on the same algorithm but using a different large search pattern, shown in Figure 2.9 (c).

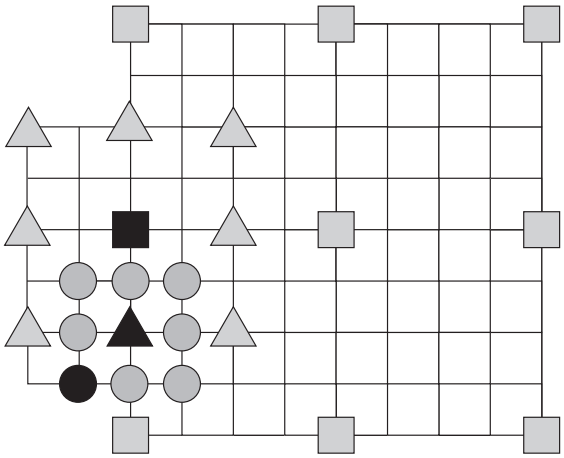


Figure 2.8: Three-step search motion estimation.

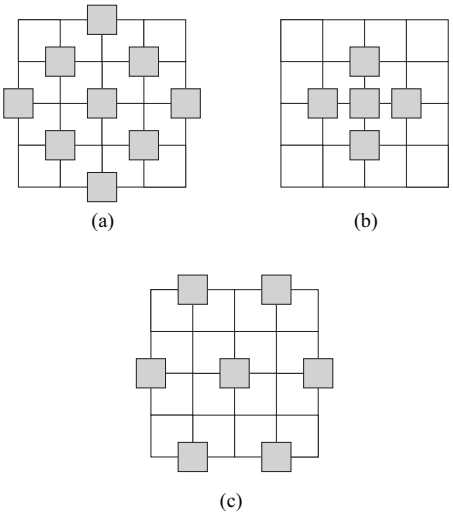



Figure 2.9: Diamond and hexagon search patterns.

A very successful class of algorithms has been proposed based on both techniques of using the motion information extracted from neighbouring blocks and using search patterns, referred to as Enhanced Predictive Zonal Search (EPZS) [38]. These algorithms work by defining a list of candidate motion vectors (the predictors), using motion vectors found in neighbouring blocks or determined in other ways. The predictors are tested to compute an initial solution, and this is then refined using pattern searches such as hexagon search.

Exhaustive search returns the solution at minimum distortion only under the assumption that the reference frame is left unchanged. On the contrary, already during the initial developments of video coding technology it was observed that better prediction accuracy may be obtained manipulating the reference frame prior to performing motion compensation. As a result sub-pixel motion estimation was proposed [39]. The idea is to extend the set of motion vector candidates allowing motion vectors to assume fractional values. The reference frame is interpolated to a desired sub-pixel precision by means of appropriate filters, and the motion search is performed on these interpolated samples. Sub-pixel motion estimation algorithms are used in the majority of video coding applications due to their very high efficiency as illustrated in the rest of this thesis. Nonetheless such algorithms are particularly challenging especially in terms of computational complexity and resource requirements. For instance in the simplest case of half-pixel precision, the horizontal and vertical sizes of the reference frame are doubled. Assuming a full search algorithm using a given search window, motion estimation would require testing four times the motion vector candidates than those required when only considering integer-precision. Moreover, the interpolated samples need to be available to encoder and decoder, with possibly very high demands in terms of storage resources.

Low complexity sub-pixel motion estimation has been extensively investigated in the past for these reasons. Also in the case of sub-pixel motion estimation the convexity of the distortion error is a common assumption for developing fast algorithms. Fractional solutions are assumed to be located in the surroundings of the integer-precision solution.

A fast quarter-precision sub-pixel algorithm was proposed, referred to as hierarchical search, which is a simple extension of the three step search algorithm to the fractional case. Also a fast two-step algorithm for quarter-precision motion estimation [40] was proposed based on a first search among only four half-precision locations, and a second search in the surrounding of the two best solutions. A method based on using the motion information of previously encoded blocks was also proposed [41]. Some techniques have been proposed with the goal of reducing complexity of interpolation. A class of algorithms was proposed [42] [43] [44] to perform interpolation and motion search in a single step, using models for interpolating the distortion values at integer precision.

In order to increase prediction accuracy the motion search can be extended to span across multiple reference frames. More candidates are tested which means that possibly a better solution can be obtained, at the cost of higher computational complexity. Clearly the motion information needs to be extended to include an index to identify which reference frame should be used for motion compensation. Recent video coding schemes extend this concept to allow frames to be encoded in a different order than their actual display order. In this way frames at a future time instant than the current frame can be also used as references. Bidirectional prediction [45] was introduced as a further extension of this idea. Two previously encoded frames (typically one previous and one future frame in display order) are concurrently considered. Two sets of motion vector candidates are tested (one per reference frame), where a candidate from each list is tested and used to extract a prediction from the corresponding reference. The two prediction blocks are combined together to predict the content of the current block, as illustrated in Figure 2.10. Two motion vectors and two reference indexes (one per reference frame) need to be transmitted for each block when using bidirectional prediction. Note that in schemes in which bidirectional prediction is allowed, conventional motion estimation performed on a single reference frame is sometimes denoted as unidirectional motion estimation.

Several other methods have been proposed to improve inter-prediction. Global motion

estimation [46] was proposed, which consists in combining multiple references to obtain a new frame, before encoding the current frame. This new frame is added to the list of possible references and used during the motion search. A work on luminance transform based on brightness compensation has been presented [47] as an alternative to motion estimation. Weighted prediction has been proposed [48] particularly to address scene changes and fades in video sequences based on transformation of reference frames by means of weighting factors.

### 2.2.4 Transform coding

The transform unit in a video encoder has the goal of finding a representation of the signal more suitable for compression, compacting most of the energy in a small number of parameters. The idea is that parameters that contribute with small energy may be discarded with little effects when consuming the signal. Well known ways of obtaining such a representation are the discrete cosine transform (DCT) [49] or discrete sine transform (DST), two famous members of a particular family of sinusoidal unitary transforms derived from discrete Fourier analysis. A sinusoidal unitary transform is an invertible linear transform whose kernel describes a set of complete, orthogonal discrete cosine and/or sine basis functions. The well known Karhunen-Loeve transform (KLT) [50] is

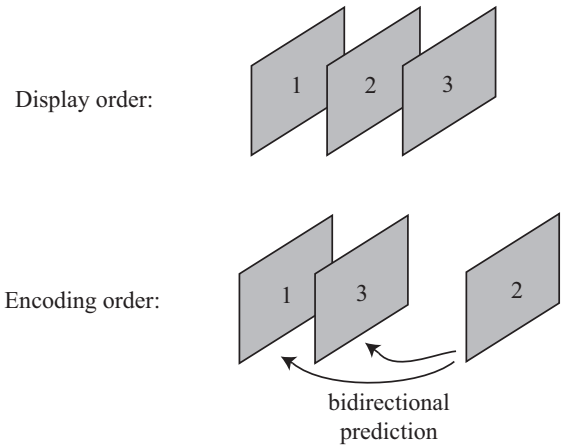



Figure 2.10: Bidirectional motion estimation.

another member of the same class of transforms. Different types of DCT and DST have been defined [51], where the two-dimensional versions of types II and III are often used in image and video coding for the forward and inverse transform respectively.

When using forward DCT, an array of $N$ samples is multiplied on the right by an $N \times N$ transform base matrix to obtain an equally sized array of $N$ values referred to as the transformed coefficients. The elements of the transform base matrix are extracted from discrete (scaled) cosinusoids at different frequencies, or:

$$Q(m,n) = k_n \cos\left(\frac{\pi(2m+1)n}{2N}\right), \tag{2.3}$$

where:

$$k_n = \begin{cases} \frac{\sqrt{2}}{2} & \text{if } n = 0 \\ 1 & \text{otherwise.} \end{cases}$$

The columns of the transform base matrix contain the $N$-points realisations of particular functions referred to as the DCT basis functions, as illustrated in Figure 2.11. The DCT has a strong decorrelating effect, namely the coefficients $\widetilde{r}(n)$, $n = 0, ..., N-1$ are less correlated one to each other than the original samples $r(m)$, $m = 0, ..., N-1$ prior to transformation. For instance consider a 2-point DCT and a signal $r(0), r(1)$. From Equation 2.3, $\widetilde{r}(0) = \frac{\sqrt{2}}{2}(r(0) + r(1))$ and $\widetilde{r}(1) = \frac{\sqrt{2}}{2}(r(0) - r(1))$. If $r(0) \simeq r(1)$ (the samples are highly correlated), then $\widetilde{r}(0) \simeq \sqrt{2}r(0)$ and $\widetilde{r}(1) \simeq 0$. While the original

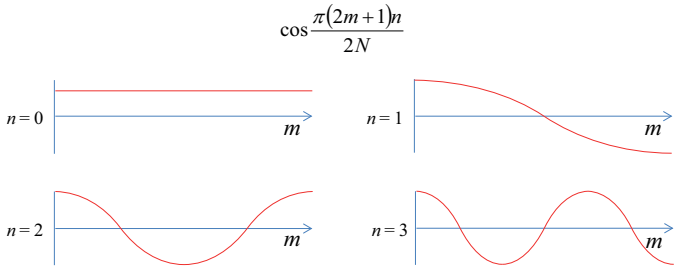



Figure 2.11: The DCT basis functions.

samples shared around half of the total energy of the signal, in the transformed signal most of the energy is compacted in the first coefficient.

Two-dimensional DCT of an $N \times N$ block of samples is defined by successively applying two stages of DCT to the rows and columns of the block respectively. Given a block $\mathbf{R}$ of samples and referring to each sample as $r(i, j)$, this corresponds to:

$$\widetilde{r}(m,n) = k_{m,n} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} r(i,j) \cos \frac{\pi(2i+1)m}{2N} \cos \frac{\pi(2j+1)n}{2N}, \tag{2.4}$$

where $k_{m,n}$ are the scaling factors, and $\widetilde{r}(m, n)$ are the transformed coefficients.

When $m = 0$ and $n = 0$ the frequency of the two cosinusoids in Equation 2.4 becomes zero and the corresponding basis function reduces to a constant. The associated transformed coefficient $\widetilde{r}(0, 0)$ is referred to as the DC coefficient. Each following value of $m$ and $n$ refers to an increasingly higher frequency component. The corresponding coefficients are referred to as AC coefficients.

Some alternative transforms to DCT have been investigated for video coding applications. Despite its decorrelating properties and ease of computation, DCT may ne bot optimal particularly in some cases such as when coding blocks of intra-predicted residuals [52]. A study on the compression performance provided by different transforms in the case of angular intra prediction was presented [53], showing a correlation between angular directions of the prediction and optimal transform. Similarly, another study on the optimality of the DCT transform making use of Gaussian Markov random field models [54] concluded that DCT is indeed not optimal for intra-predicted residual signals. Wavelets have also been extensively investigated in the context of video coding.

In order to take advantage of the representation provided by the transform this is followed by quantisation. This is performed dividing each coefficient by a certain step and rounding to the nearest integer. The quantised coefficients can assume a smaller

range of values and therefore require less bits to be encoded. Quantisation is a non-reversible process which involves a loss of information: the larger the quantisation step, the smaller the number of allowed values, resulting in lower bitrates but also higher losses of information. Due to the fact that most of the energy of the signal should be compacted to low frequencies, in theory coefficients at high frequencies can be quantised with larger steps. This may be achieved by means of so called quantisation matrices, namely matrices containing different quantisation steps for each frequency component. As a result of the large quantisation steps, coefficients at high frequencies may be completely discarded and replaced with zero-valued components. Note that typical video coding schemes allow the expected quality of the signal to be selected prior to the encoding, by appropriately manipulating and scaling the quantisation matrices.

### 2.2.5 A Brief History of Video Coding Standards

The first documented international video standard (probably the first standard for any visual content) is the H.120 [55], ratified in 1984 by the CCITT (Comite Consultatif International Telephonique et Telegraphique), which at the times was part of the International Telecommunication Union (ITU), an organization whose intent is to produce standards for telecommunications. Pixels in H.120 are not grouped in blocks but are coded independently. H.120 was poorly received by industry and public, and few implementations appeared in the market.

Following from such a negative feedback the CCITT formed a group of experts dedicated exclusively to the definition of its successor. The development followed a collaborative approach based on comparison of competing proposals, resulting in 1988 in the ratification of the first block-based hybrid codec as the H.261 standard [11]. The basic processing unit of H.261 is called macroblock. Each $16 \times 16$ macroblock is coded using integer-precision motion estimation, DCT and quantisation.

As a collaborative approach between the Joint Photographic Experts Group (JPEG)

and the CCITT, the Moving Picture Experts Group (MPEG) was established in January 1988, as an organization completely dedicated to developing standards for video signals. The first standard ratified by the group was the MPEG-1 [56], approved in 1993. MPEG-1 is largely based on H.261 but includes several novel techniques such as bidirectional prediction and half-pixel precision motion estimation.

In 1993 CCITT was renamed to ITU-T (ITU Telecommunication Sector). While the DVD format had just been introduced in the market, neither H.261 or MPEG-1 were considered suitable for the relatively high qualities targeted by the format. ITU-T and MPEG worked together with the goal of quickly developing a successor, and the H.262 or MPEG-2 Video standard (as named by the ITU-T or MPEG respectively) was ratified as a result [57] in 1994. Global motion estimation was introduced in the standard along with many other small enhancements particularly in the inter-prediction module. Many amendments were later introduced to the standard; MPEG-2 Video is still supported nowadays in some commercial applications.

H.263 was ratified in 1996 [58], specifically to target low bitrate applications. It included novelties such as variable block size partitioning, three dimensional variable length coding and motion vector prediction. The H.263+ extension was ratified in 1998 to extend support for chroma subsampling formats other than $4:2:0$ and include error resilience.

Apart from efforts of ITU-T and MPEG, other standards have also been proposed. Microsoft commercialised several of such products. Already in 1999 Windows Media Video 7 (WMV-7) was introduced (a proprietary re-implementation of MPEG-2 Video). A fully independent standard was later commercialised in 2005 as WMV-9, reportedly capable of achieving comparable compression efficiency with respect to its competitors [59]. WMV-9 includes novel techniques such as for variable block size transform, particularly interesting because similar techniques would later be adopted also by ITU-T and MPEG standards. WMV-9 was later included as part of a broader project under the name of VC-1 [60], standardised by the Society of Motion Picture and Television Engi-

neers (SMPTE) in 2006. VC-1 and WMV-9 have been adopted by some online video distribution services and are widely supported by Microsoft products.

The British Broadcasting Corporation (BBC) Research and Development department internally developed a royalty free and open source wavelet-based video coding standard called Dirac [61]. The standard specifications were completed in 2007 and include a version (Dirac Pro) which only defines the intra-prediction subset of specifications as a tool for professionals in the video production industry. Dirac Pro was approved for standardisation by the Society of Motion Picture and Television Engineers (SMPTE) in 2010 under the name of VC-2 [62].

Finally Google is also active in developing video coding standards. The VP8 [63] standard was originally developed in 2008 by On2 Technologies (a small company involved in video compression). On2 was later acquired by Google. VP8 makes use of very similar methods as the ITU-T and MPEG standards. It also introduces some novel techniques such as a method to artificially construct reference frames using previously coded information, and low-complexity fractional interpolation for sub-pixel motion estimation. A successor of VP8 called VP9 was recently presented in 2013 [64]. At the time of writing the performance of this standard in comparison with its competitors is still unclear due to the inconsistencies of results presented so far [65] [66].

## 2.3 H.264/AVC

### 2.3.1 General Information and Syntax Elements

The H.264/AVC (Advanced Video Coding) standard [67], referred to as AVC in the rest of this thesis, is at the time of writing one of the most widely used international video coding standards for consumer and professional applications. The impact of AVC on the related industry is enormous. Many countries use AVC for at least part of their digital terrestrial television services, including UK where it was adopted for encoding

the Freeview HD digital television signal via DVB-T2 [68]. It is also extensively used in satellite television especially for coding HD signals, including for instance Sky TV and BBC HD via DVB-S [69] in the UK. Moreover, AVC is nowadays widely used for internet streaming in services such as Youtube, iTunes Video or Vimeo, and it was chosen as the exclusive video coding standard for the BBC iPlayer since 2006. Modern video camcorders from a broad range of manifacturers including Canon, Sony, Nikon or Samsung, encode video sequences in real time using the AVC standard. All Blu-Ray players must be able to decode AVC bitstreams [70].

AVC was jointly developed by the Video Coding Experts Group (VCEG) of the ITU-T organisation, and the MPEG group, in the form of a collaborative team under the name of Joint Video Team (JVT). JVT worked specifically with the goal of providing the best possible framework for video compression, and finally ratified the standard in 2003. At that time AVC was already reportedly able to improve the coding performance by a factor of two over its predecessor MPEG-2-Video. While the first version of the standard only supported $4:2:0$ chroma subsampling and 8-bit data representation, the JVT continued working on the project and later developed the Fidelity Range extension (FRext), an improved version capable of supporting different chroma subsamplings and higher bitdepths. Compared to previous standardisation efforts, AVC provides improvements in almost all aspects of video compression, from prediction accuracy to coding efficiency, robustness, and error resiliency [71].

As with previous video coding standards, only the decoder and the bitstream structure (also referred to as the syntax) are specified in the standard text specifications. The encoder can be implemented according to specific needs to produce a bitstream compliant with the specifications. Note that along with the development of the standard, a reference software has also been developed (mostly by the JVT members), referred to as the Joint Model (JM) reference software [72]. JM was created with the main goal of helping the JVT members in testing new proposed methods, showing the results of particular techniques, and measuring their compression efficiency against an established

benchmark. The software is available following the open source model and includes both an AVC decoder and encoder. Most of the work related with AVC in this thesis was developed using this software.

AVC follows the hybrid model [73]. The input data is processed partitioning the video sequence in coding entities called slices. Each slice is further partitioned in square macroblocks formed of a fixed number of $16 \times 16$ luma samples, along with the corresponding chroma samples according to the chroma subsampling being used. A slice is in fact defined as a collection of an integer number of macroblocks. While in theory a frame in the sequence could be coded as multitude of slices, in practice it is common to code each frame as exactly one slice. For this reason the terms"frame" and "slice" are used without distinction in the rest of this thesis unless explicitly specified. A macroblock is the main building block of AVC. Each macroblock is processed as in the hybrid model: the encoder generates a prediction which is then subtracted to the original block to produce the residual samples. These are input to the transform unit where they are transformed and quantised, before being input to the entropy coder for encoding in the bitstream. AVC also includes an additional step which involves the refinement of the decoded signal by means of filters, appropriately defined to further improve the reconstruction quality.

The structure of the syntax of a slice coded using AVC is illustrated in Figure 2.12. Each slice is formed of slice header and slice data, as illustrated in the left of the figure. Each slice can be processed following different schemes depending on predefined slice types. The type is coded for each slice in the slice header, along with some information regarding the picture from which the slice is extracted, and possibly additional data.

There are five possible slice types defined in AVC:

1. I slices only contain macroblocks that are coded using intra-prediction.

2. P slices may contain macroblocks coded using either intra-prediction or unidirectional inter-prediction with reference frames extracted from only one list (more

details on this in the following of this chapter).

3. B slices may contain all types of supported macroblocks.

4. SP and SI slices are particular versions of respectively P or I slices. These are not of interest for the purpose of this thesis.

The slice data consists of a sequence of coded macroblocks. AVC supports a predefined number of macroblock types, which define the way the prediction is computed for the related samples. The macroblock type and the information required to compute the prediction are encoded at the beginning of each macroblock in a portion of the syntax referred to as macroblock information (MB info). The rest of the syntax for a macroblock possibly contains the residual data. Note that the encoder might decide that there is no need to transmit any residual information (referred to as SKIP mode, as illustrated later in this chapter). In this case only the MB info is encoded for a macroblock as shown in the case of the second macroblock in the right of Figure 2.12.

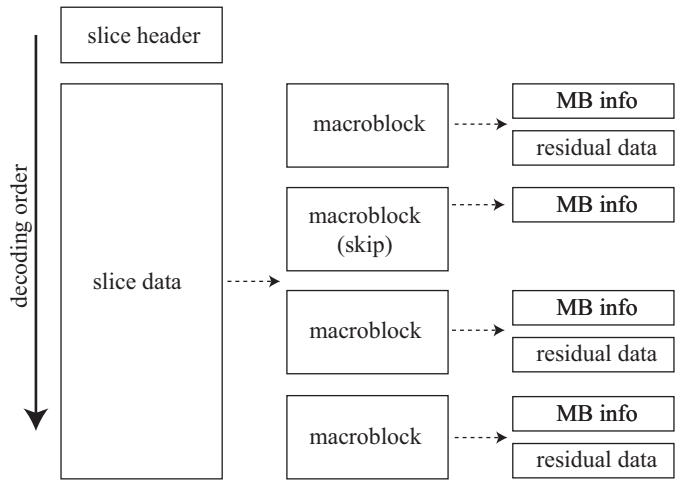



Figure 2.12: Slice syntax elements in AVC.

### 2.3.2 Intra-prediction in AVC

Intra-prediction is the only option when coding macroblocks in I slices. I slices appear for instance when coding the first frame in the sequence, or when a refresh is needed during the coding to remove any dependency from previous parts of the bitstream (this is usually referred to as a "random access" point). Moreover in some cases intra-prediction might provide better results than inter-prediction at generally lower computational costs, which is why it is usually tested also when coding P or B slices.

AVC supports three types of intra-predicted macroblocks, namely Intra-$4 \times 4$, Intra-$16 \times 16$, and I_PCM [74]. When using Intra-$16 \times 16$, all luma samples in the macroblock are predicted together using neighbouring samples. Up to 33 samples are used for the prediction (16 from the top and left side respectively, plus the top-left corner sample). Four Intra-$16 \times 16$ modes are supported (denoted as mode 0 to mode 3). Mode 0 (vertical intra-prediction), mode 1 (horizontal intra-prediction) and mode 2 (DC prediction) follow the methods already detailed in this chapter. Mode 3, referred to as planar (or plane) intra-prediction, is a new method introduced in AVC. Planar mode consists in successively interpolating boundary samples to obtain the plan that best approximates such samples. Two examples of Intra-$16 \times 16$ prediction blocks are illustrated in Figure 2.13.

When using Intra-$4 \times 4$, the macroblock is sub-partitioned in 16 sub-blocks of $4 \times 4$ luma samples independently predicted. This provides better efficiency when coding highly textured areas of a frame. The sub-blocks are coded from left to right, top to

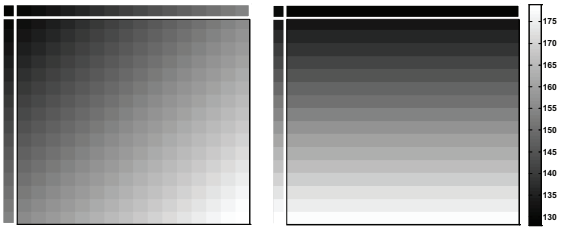



Figure 2.13: Intra-prediction in AVC.

bottom as in Figure 2.14 (a). Each sub-block is predicted using up to 13 samples (8 from the top side, 4 from the left side, plus the top-left corner sample) as shown in Figure 2.14 (b). Note that when coding sub-blocks numbered from 5 to 16 in the figure, some or all of the prediction samples are extracted from previously encoded sub-blocks within the currently encoded macroblock. Nine Intra-$4 \times 4$ modes are supported, namely 8 angular modes and DC prediction, as illustrated in Figure 2.14 (c). Mode 0 (vertical intra-prediction), mode 1 (horizontal intra-prediction) and mode 4 (diagonal down-right intra-prediction) follow the methods already detailed in this chapter. The remaining modes 3, 5, 6, 7 and 8 involve the interpolation of reference samples. In particular denote with $p(i, j)$ where $i, j = 0, ..., 3$ the predicted samples, denote with $p(i, -1)$ where $i = 0, ..., 3$ the reference samples in the left side, with $p(-1, j)$ where $j = 0, ..., 7$ the reference samples in the top side and denote with $p(-1, -1)$ the reference sample in the top-left corner as in Figure 2.14 (b). For $i = 0$ or $i = 2$:

$$p(i, j) = \frac{p(-1, j + i/2) + p(-1, j + i/2 + 1)}{2}.$$

For $i = 1$ or $i = 3$:

$$p(i, j) = \frac{p(-1, j + (i-1)/2) + 2p(-1, j + (i-1)/2 + 1) + p(-1, j + (i-1)/2 + 2)}{4}.$$

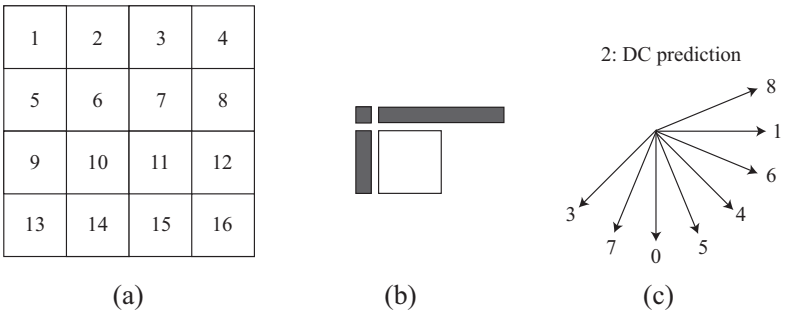



Figure 2.14: Intra-$4 \times 4$ prediction in AVC. Order of prediction of sub-blocks (a). Reference samples for each sub-block (b). Intra-prediction modes (c).

A method was proposed [75], in which sub-blocks in Intra-$4 \times 4$ modes are coded in a different order than in conventional AVC to allow for more reference samples to be used during the prediction. Note finally that chroma samples are predicted for the entire macroblock at once regardless of the intra-prediction macroblock type, following similar techniques as luma samples.

Eventually the I_PCM mode is also considered in AVC. When using I_PCM, prediction and transform unit are completely skipped during the encoding, and input samples are directly entropy coded in the bitstream. This is considered to address unexpected anomalous content in particular regions of a frame, to avoid situations in which the total amount of bits required to encode prediction and residual data exceeds the bitrates necessary to directly encode the original samples.

### 2.3.3 Inter-prediction in AVC

Inter-prediction is supported in P or B slices. Inter-predicted macroblocks require information extracted from one or more previously coded reference frames. AVC introduces an efficient method to flexibly adapt the size of the partitions used for inter-prediction, depending on the macroblock content. In particular each $16 \times 16$ array of luma samples may be coded using one of four modes: mode $16 \times 16$ corresponds to computing a single prediction for the whole macroblock; in mode $16 \times 8$ and $8 \times 16$ the macroblock is split in two identical partitions (in the vertical or horizontal direction respectively) that are independently predicted. When using mode $8 \times 8$ the macroblock is split in four blocks of the same size. In this case, each $8 \times 8$ block may be further sub-partitioned using one of four sub-modes: sub-mode $8 \times 8$ corresponds to computing a single prediction for the whole block at once; in sub-modes $8 \times 4$, $4 \times 8$ and $4 \times 4$ the block is further partitioned in sub-blocks. A graphical representation of the inter-prediction modes and sub-modes used in AVC is shown in Figure 2.15.

Each block or sub-block is independently predicted by means of motion estimation

and motion compensation using one or more previously encoded reference frames. Up to two lists of reference frames (referred to as list 0 and list 1) may be considered depending on the slice type. Macroblocks in P slices can only consider reference frames included in list 0; unidirectional prediction is considered when predicting these macroblocks. Macroblocks in B slices may consider reference frames in both lists; either one reference is extracted from one of the lists to perform unidirectional prediction, or two frames are extracted (one per list) to perform bidirectional prediction. The lists are populated with previously encoded frames following a complex algorithm. Both lists have a limited number of available slots. In theory list 0 should contain past reference frames (whose temporal index is smaller than the current index) and list 1 should contain future frames. In practice in case the number of previous or past frames is too large to fit within the corresponding list, frames may be included in the other list regardless of their temporal index.

AVC allows fractional motion estimation with quarter-pixel precision (for the luma component). Typically each reference frame is interpolated prior to including the frame in one of the lists. The interpolation is based on two successive steps, to obtain the half-precision and quarter-precision samples respectively.

Half-precision samples are obtained by means of a 6-tap filter in two passes. Formally, denote as $p(m, n)$ the samples in the reference frame where integer values of $m$ and $n$ correspond to integer-located samples. In the first pass only half-precision samples

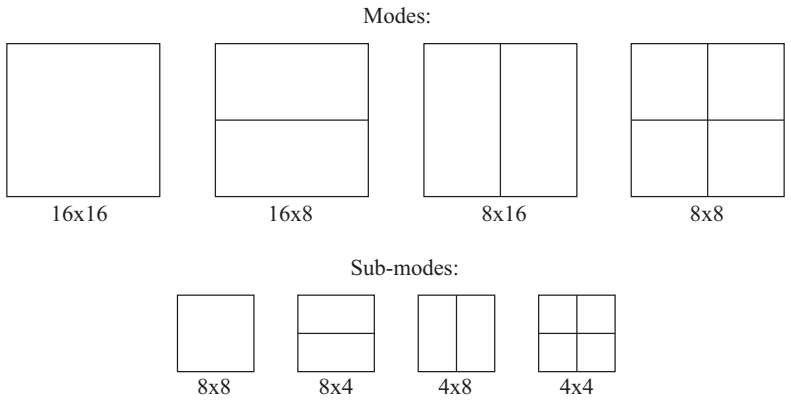



Figure 2.15: Inter-prediction modes in AVC.

located between integer-precision samples are computed (namely only one coordinate $m$ or $n$ is fractional), as illustrated in Figure 2.16. If $m$ is fractional:

$$p(m,n) = \frac{p(m-2.5,n) - 5p(m-1.5,n) + 20p(m-0.5,n) + 20p(m+0.5,n) - 5p(m+1.5,n) + p(m+2.5,n)}{32},$$

otherwise if $n$ is fractional:

$$p(m,n) = \frac{p(m,n-2.5) - 5p(m,n-1.5) + 20p(m,n-0.5) + 20p(m,n+0.5) - 5p(m,n+1.5) + p(m,n+2.5)}{32}.$$

The remaining half-precision samples are computed in the second pass using the same filter on the samples computed in the first pass.

Quarter-precision samples are computed by means of linear interpolation of two integer-precision or half-precision samples, also in two passes following a similar process. For example in the first pass, if only $m$ has quarter-precision accuracy:

$$p(m,n) = \frac{[p(m-0.25,n) + p(m+0.25,n)]}{2}.$$

A method was proposed to extend the interpolation scheme used in AVC by means of generalized interpolation [76], obtaining some gains in compression efficiency at the cost of higher computational complexity.

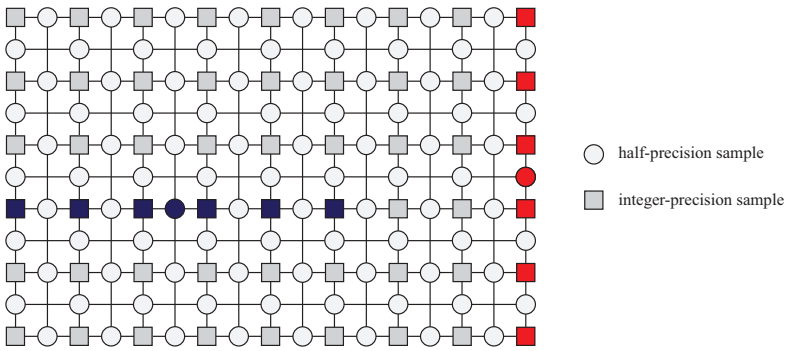



Figure 2.16: First pass of half-precision interpolation in AVC.

The motion information necessary to inter-predict each block or sub-block must be signalled in the bitstream so that motion compensation can be applied also at the decoder side. Especially in case of high granularity of the macroblock partitioning (for instance when considering many small sub-blocks), encoding this motion information might require a very large number of bits. In order to mitigate this problem AVC makes use of an efficient method referred to as motion vector prediction, which attempts at exploiting the correlation between neighbouring blocks (or sub-blocks) to reduce the costs of transmission of motion vector components. Under the assumption that motion vectors found for neighbouring blocks (or sub-blocks) might be similar with the optimal motion vector to be used in the current block, a good estimate of this motion vector may be obtained using these previously computed motion vectors. The motion vector prediction is subtracted to the original motion vector components to obtain the motion vector difference, which is then encoded in the bitstream. The components of the motion vector difference should be smaller than the original components and therefore require less bits to be encoded.

The motion vector prediction is derived in AVC using a list of motion vector prediction candidates. The partition on the left of the current partition is considered (if there are more than one partition, the top-most one is considered). The corresponding motion vector is included in the list as $MV_a$. Similarly the partition on top of the current partition is considered (if there are more than one partition, the left-most one is considered). The corresponding motion vector is included in the list as $MV_b$. Finally the partition at the top-right corner of the current partition is considered. The corresponding motion vector is included in the list as $MV_c$. If $MV_c$ is not available, $MV_d$ is included in the list instead, from the partition at the top-left corner of the current partition. These are illustrated in Figure 2.17. The median of the elements in the list is then used as motion vector prediction for the current block.

While the motion estimation algorithm is not normative in the standard, the JM reference software includes several algorithms such as full search, or fast search using

EPZS. Finally it should also be mentioned that weighted prediction [77] is included in AVC. When using weighted prediction, the reference frames are transformed using a weighting factor and additive offset. Two weighted prediction modes are supported, referred to as explicit mode and implicit mode. The implicit mode is mainly used to improve bidirectional motion estimation; instead of using the plain average of the two reference frames used for the prediction, these are scaled depending on the temporal distance with the frame currently being encoded. Conversely in the explicit mode, the weighting factor is computed at the encoder side for each reference frame before including the reference frames in the lists, and transmitted in the slice header. The method was extended to include a prediction of the weighting parameters [78].

AVC includes two additional inter-prediction modes referred to as SKIP and direct mode that are particularly efficient in the case of high temporal correlation among neighbouring frames (e.g. in case of static content spanning across a number of successive frames). SKIP mode is used in P slices and direct mode is used in B slices; while there are some differences, the two modes are based on the same concept. In the presence of high temporal redundancy three things are likely to happen in a macroblock: (i) the content of the entire macroblock can be efficiently coded using a single prediction; (ii) the optimal motion vector can be very similar to those found in neighbouring blocks; (iii) high prediction accuracy can be expected resulting in very low residual samples. SKIP mode takes all these factors into consideration: (i) prediction and reconstruction are computed for the entire $16 \times 16$ macroblock; (ii) no motion information is transmitted, instead the

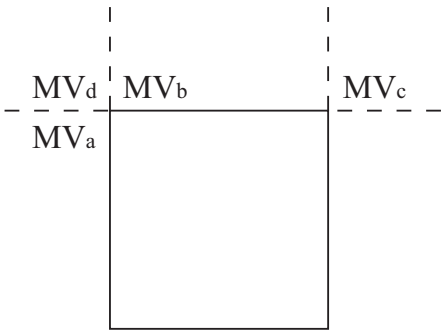



Figure 2.17: Motion vectors used for motion vector prediction in AVC.

motion vector prediction is used directly to compute motion compensation; (iii) no residual samples are encoded, transform and entropy coding are completely skipped and the motion compensated prediction is used as reconstruction for the macroblock. A single binary flag (to signal that SKIP is used) is needed to completely represent a skipped macroblock. Especially in signals containing large amounts of static content, very high compression efficiency can be expected as a result of appropriately using the SKIP mode.

### 2.3.4 Transform and Other Tools in AVC

The residual samples are computed as the difference between original macroblock and prediction before being transformed. The transform is not computed for the entire $16 \times 16$ samples at once, but instead the macroblock is split in $4 \times 4$ blocks that are independently transformed. This additional partitioning has the goal of isolating and capturing sharp transitions in the signal, therefore possibly reducing compression artefacts.

The entries in the DCT transform base matrix are irrational numbers, and rounding is necessary before these can be stored in digital representation. In order to reduce the norm of the transform base matrix and consequently allow for a lower dynamic range of the variables output by the transform stages, a different base matrix is used in AVC [79] that shares some of the DCT properties but also has additional advantages. The transform was derived again rounding the elements of a DCT base matrix, but these are further adjusted to obtain a transform that is simpler to implement, and which allows 16-bit data representation of the output variables. In particular:

$$Q = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix},$$

is used in AVC followed by a scaling matrix appropriately defined.

AVC makes also use of an additional step, referred to as hierarchical transform, to reduce the correlation among different $4 \times 4$ blocks within a macroblock in the case of static content (such as in Intra-$16 \times 16$ macroblocks or chroma components). In these cases the similarity among $4 \times 4$ blocks within a macroblock can be exploited to achieve higher compression efficiency. The DC coefficients of each $4 \times 4$ block are collected in a matrix that is further transformed. To limit the complexity of this additional step, Hadamard transform is used instad of DCT, defined by the following base matrix:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}.$$

Finally it should be noted that different transform techniques have been investigated in the context of AVC. Wavelets and lapped transforms have been used as an alternative to conventional intra-prediction and DCT-based transform stages [80]. Also, a method to adapt the transform kernel function to the content currently being coded has recently been proposed [81].

After transform, the coefficients are quantised. The quantiser step is determined by the value of a quantisation parameter (QP), which may be set at a sequence, frame or macroblock level depending on the coding configuration. High values of the QP correspond to low expected quality of the decoded signal. Typical applications which require medium quality of the decoded signal usually consider values of the QP between 22 (relatively high quality) to 37 (low quality at relatively low bitrates). Finally the quantised coefficients are input to the entropy coder. Some details on the entropy coding methods used in AVC can be found in Appendix A.

As an effect of partitioning the frames in macroblocks, blocking artefacts may appear

when decoding an AVC bitstream especially at lower qualities (i.e. when using high values of the QP). To partially address this problem AVC allows the frames to be filtered by an appropriately designed adaptive in-loop filter. Details on this filter can be found in the literature [82] and are out of the scope of this thesis.

## 2.4 H.265/HEVC

### 2.4.1 General Information

AVC was developed at a time when UHD resolutions for consumer applications were unimaginable, a great portion of the content was consumed by means of physical media such as DVD video, access to broadband services was still limited, and only a few mobile devices were capable of reproducing multimedia signals. Thanks to new technological breakthroughs and a fast adapting market, 10 years later everything changed. Widespread access to broadband internet is nowadays common even in mobile devices such as smartphones or tablets. New technological breakthroughs pushed the market towards higher resolution devices with definitions higher than 1080p, with the consequent demand for content in UHD formats. Also, the number of applications that need efficient video coding technology is quickly growing. Video content distribution over the internet, real-time display mirroring, high-definition video capturing by small devices with limited amount of internal storage, are all examples of applications that had a very limited scope only a few years ago but are instead relevant in the current scenario.

The ITU-T VCEG issued a call for proposals in 2009 [83] to address these emerging needs for new video coding technologies. As a response to this call another collaborative team between the ITU-T and MPEG group was established under the name of Joint Collaborative Team on Video Coding (JCT-VC). Following the successful collaboration that led to AVC and its extensions, the team committed to the development of a next-generation video coding standard with the goal of consistently reducing the

bitrates required when coding sequences, under a broader set of conditions as compared to AVC. The standardisation process was again based on the proposal and validation of competitive technologies to select only the most efficient tools. A first version of the standard was finally approved in January 2013 under the name of H.265/HEVC (High Efficiency Video Coding) [84], referred to as HEVC in the rest of this thesis.

Throughout the entire standardisation process the JCT-VC members developed, updated and maintained a reference software with the intent of providing a reference during the validation of proposed tools and also of assisting potential users in understanding the standard architecture. The software, referred to as HEVC Test Model (HM) [85], provides both a standard-compliant decoder and an example encoder. HM is available under the open source model and it has been used in this thesis as the basis for implementing all the techniques and algorithms proposed in the context of HEVC.

HEVC is based on a very similar architecture to that of its predecessor [86] and follows the block-based hybrid model. On the other hand almost all aspects of the coding process are improved and optimised, and as a result the standard is reportedly capable of achieving in average more than 50% higher efficiency than AVC under standard conditions [87]. Interestingly, HEVC is also considerably more efficient than state-of-the-art standards for still image coding, for instance it is reportedly in average 44% more efficient than JPEG2000 [88].

### 2.4.2 Access Configurations and Coding Units

Similar to AVC, a video sequence is encoded in HEVC as a sequence of slices. Three slice types are supported, namely I, P or B slices, where the characteristics of each slice type are directly inherited from the equivalent in AVC. The order of coding of frames may be different than the display order depending on the access configuration. Three access configurations are particularly relevant in HEVC referred to as low-delay, random access and all intra. In the low-delay configuration the frames are coded in display order.

Only frames from temporally past instants can be used as references for inter-prediction. This is useful in those situations where the signal needs to be encoded, transmitted and decoded in the smallest amount of time (for instance for real-time broadcasting). Conversely, the random-access configuration makes use of a more complex coding order where frames are encoded in groups, each group starting with a random access point (RAP) frame. A decoder can start decoding a bitstream from any RAP frame, which means that the frames after a RAP frame cannot depend on the content in any frame before this RAP frame. Typically RAP frames correspond to I slices. Random access configuration is in general expected to provide better compression efficiency than low-delay configuration. Finally in the all intra configuration, all frames are coded as I slices. This is useful in all situations when no inter-frame dependencies are allowed.

Each slice is coded as a sequence of blocks. While AVC makes use of a fixed partitioning in macroblocks of $16 \times 16$ luma samples, one of the key factors responsible for the high efficiency of HEVC is the flexible way in which each slice can be partitioned. The main entity responsible for this partitioning is the Coding Tree Unit (CTU). A CTU is a syntax structure that identifies a square area of $N \times N$ luma samples in the frame and determines how this area is partitioned in possibly smaller blocks. HEVC allows a maximum size $N = 64$. In particular a CTU is formed of one or more split flags and a corresponding amount of square blocks of variable sizes referred to as Coding Units (CUs). Each CTU starts with a split flag, namely a binary flag that determines whether the original $N \times N$ block is coded as a single largest CU, or if the largest CU is split in four smaller CUs each of $N/2 \times N/2$ luma samples. In the second case, each CU is associated another split flag, which determines whether it is coded as a whole or it is further split in four $N/4 \times N/4$ blocks, and so on. Each CU is formed of a header (containing all parameters that are transmitted once per each CU) and some additional data (which also contains the residuals). The syntax structure of a typical HEVC slice is illustrated in Figure 2.18.

Each CU inside the CTU is assigned a depth depending on its size: depth 0 for the

largest $N \times N$ CU, depth 1 for $N/2 \times N/2$ CUs and so on. HEVC allows a minimum CU size of $8 \times 8$ luma samples, which means that in case $N = 64$ the CTU can split the original largest CU into CUs up to a maximum depth of 3. Two example CU partitioning are shown in Figure 2.19.

The fact that each frame can be partitioned in blocks of variable size possibly larger than the size of macroblocks used in AVC is crucial to the efficiency of HEVC. In fact restricting the maximum CU size to $N = 16$ (the same size as that of the macroblocks

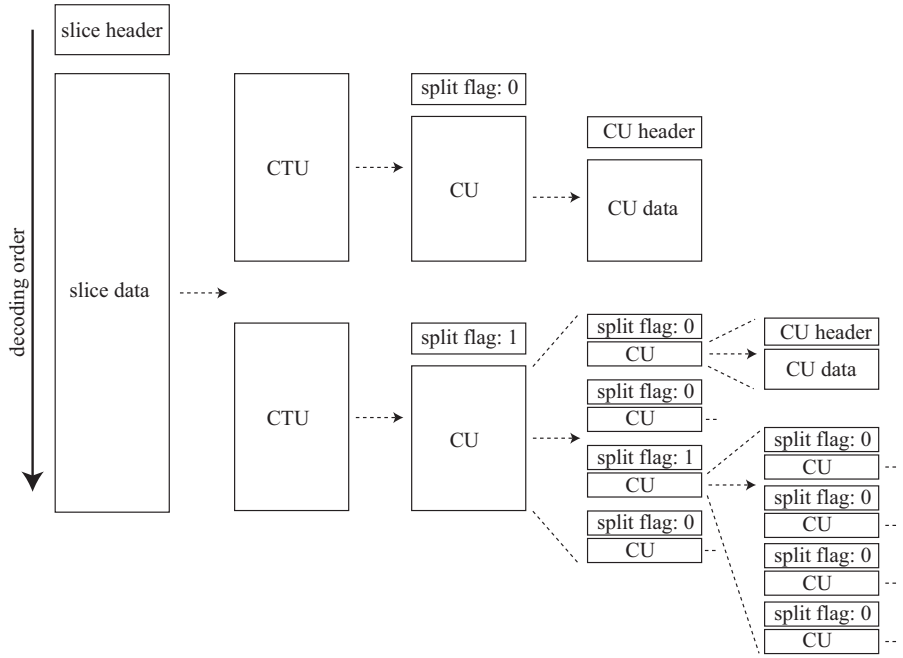



Figure 2.18: Slice syntax elements in HEVC.

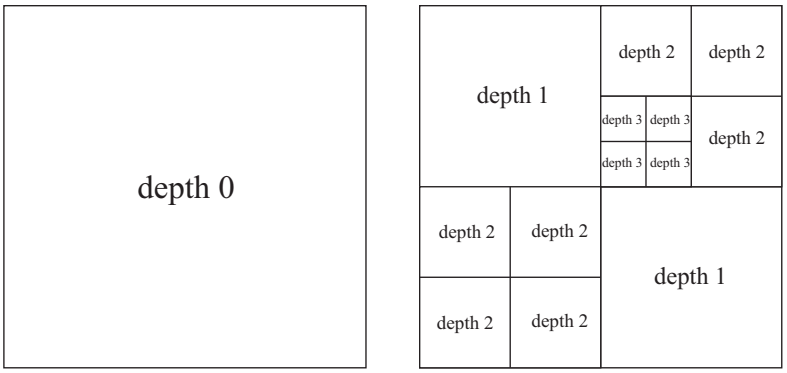



Figure 2.19: CU partitioning in HEVC.

in AVC) considerably affects the performance of HEVC [89]. A decrease in performance of up to 30% in BD-rates was reported in some cases when imposing such a restriction during the encoding.

### 2.4.3 Intra-prediction in HEVC

The block of samples considered for prediction in HEVC is referred to as prediction unit (PU). In theory the standard would allow each CU to be intra-predicted either using one single PU of the same size as the entire CU, or further partitioning the CU in four smaller PUs. These two modes are referred to in HEVC as mode Intra-$2N \times 2N$ and Intra-$N \times N$ respectively [90]. In practice the flexible CTU structure already allows each CU to identify a square region of variable size as small as $8 \times 8$ samples. For this reason only $8 \times 8$ CUs can be coded using the Intra-$N \times N$ mode (predicting four $4 \times 4$ PUs independently). All other CUs can only be predicted with mode Intra-$2N \times 2N$.

Similarly to AVC three types of intra-prediction are supported, namely DC prediction, planar prediction and angular prediction. In the latter case, up to 33 angular directions are allowed for the luma component. This is considerably more than the number of modes available in AVC, allowing for much greater prediction accuracy in a larger variety of conditions. The list of available intra-prediction modes is shown in Figure 2.20 (a). Note that the arrows depicted in the figure are only for illustrative purposes and do not accurately show the directionality of the prediction: in fact, a denser distribution of directions is considered towards the horizontal and vertical directions, and a sparser distribution of directions is considered towards diagonal directions. Also, dashed arrows in the figure correspond to angular predictions that involve interpolation of reference samples; conversely continuous arrows correspond to modes that involve simply copying reference samples in the predicted block, as illustrated in the following of this subsection.

The content of a PU is predicted using one intra-prediction mode selected from the list of available modes. It is important to point out that while the same intra-prediction

mode is used to predict all samples within a PU, the process of computing the actual intra-prediction is not performed at a PU level. In fact each PU can be further partitioned in square blocks referred to as transform units (TUs) following a process that will be detailed later in this chapter. Intra-prediction is performed separately for each TU within a PU, using different reference samples. Thanks to this method and due to the particular way in which a block is partitioned in TUs, a larger number of reference samples can be used for intra-prediction. In particular each TU of $L \times L$ luma samples is predicted by means of up to $4L+1$ reference samples as shown in Figure 2.20 (b). Not all reference samples are available for all TUs; samples that are not available are either not considered, or replaced with pre-defined values. Notice that the reference samples may consist of already decoded reference samples or filtered decoded reference samples, as will be discussed in the remaining of this subsection.

When using angular prediction, the reference samples are first arranged in an array R depending on the prediction direction. In the case of horizontal directions (modes 2 to 17) this corresponds to projecting the reference samples on top of the block (namely the arrays marked as D and E in Figure 2.20 (b)) towards an extension on top of the sample in the top-left corner (marked as C in Figure 2.20 (b)), to obtain a vertical array; conversely for vertical directions (modes 18 to 34) this corresponds to projecting the reference samples on the left side of the block (marked as A and B in Figure 2.20

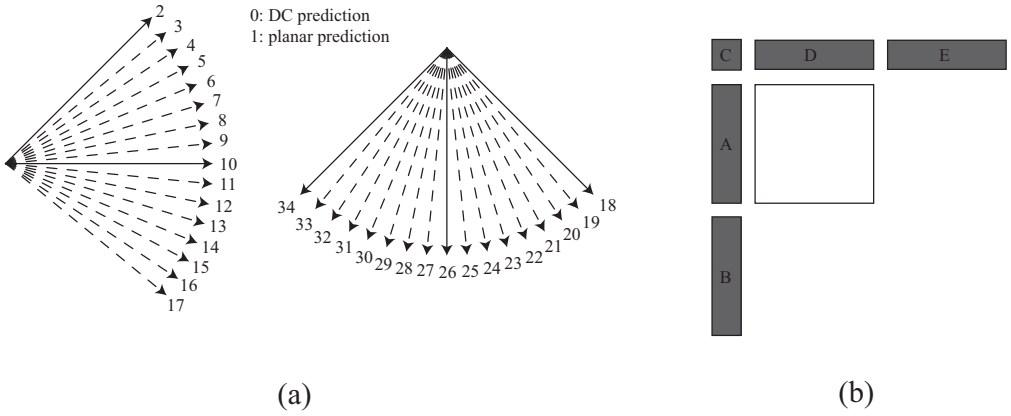



Figure 2.20: Intra-prediction modes and reference samples in HEVC.

(b))towards an extension on the left of the top-left corner to obtain a horizontal array. The elements in R are referred to as $r(t)$ where $t$ is equal to zero for the sample in the top-left corner and increases towards the right-most element (for horizontal directions) or the bottom-most element (for vertical directions) of the array. Negative values of $t$ correspond to elements in the projected portion of the array.

Also a parameter $d$ is considered that identifies the prediction direction. The parameter is allowed to assume a fixed number of possible values (the list of values can be found in the standard specifications [90]). For instance in the case of horizontal directions, these span from $d = +32$ (for mode 2) to $d = -26$ (for mode 17); $d = 0$ corresponds to mode 10 (exactly horizontal). A similar assignment is defined for vertical directions. Finally, denoting with $p(i, j)$ where $i, j = 0, ..., N-1$ the samples in the currently predicted block the prediction is computed as:

$$p(i,j) = \frac{w_j}{32} r(t) + \frac{(32 - w_j)}{32} r(t+1), \tag{2.5}$$

where all predicted samples are approximated to the nearest integers, the weighting factor is $w_j = |jd| \% [31]$, with $\% [\bullet]$ being the modulo operator, and the reference index $t$ is:

$$t = i + c,$$

where:

$$c = \left\lfloor \frac{jd}{32} \right\rfloor,$$

with $\lfloor \bullet \rfloor$ corresponding to rounding to the nearest integer smaller than its argument.

For example consider mode 11; this is an almost exactly horizontal mode with $d = -2$. First R is filled with reference samples to obtain a vertical array formed by concatenating the arrays E, D, C, A, B in this order, where samples in E and D are sorted in reverse

order; each element in R is identified with an index $t$ where $t = 0$ corresponds to the top-most element in A and negative values of $t$ correspond to elements in C, D and E. Assuming for instance that the sample $p(3, 2)$ is being predicted, using the aforementioned expressions, $c = -1$ and correspondingly $t = 2$. Such sample is predicted using the reference samples in locations $t = 2$ and $t = 3$ in the array R. Finally the weighting factor is computed as $w_2 = 4$ and the prediction is calculated as:

$$p(3, 2) = \frac{4}{32} r(2) + \frac{28}{32} r(3),$$

approximated to the nearest integer.

DC prediction is performed in HEVC in the same way as in AVC with the only difference that more reference samples may be used when computing the DC value. More interestingly, planar prediction is optimised in HEVC to adapt to the variable block size of the blocks and to avoid discontinuities at the block boundaries. Only reference samples in the arrays A and D in Figure 2.20 (b) are used for planar prediction. In particular refer to the reference samples in A as $r_A(i)$ and to the reference samples in D as $r_D(i)$ with $i = 0, ..., N-1$. Denote with $r_A = r_A(N-1)$ and $r_D = r_D(N-1)$ the bottom-most sample in A and the right-most sample in D respectively. For each sample $p(i, j)$ two linear interpolations are first computed as:

$$p_A(i, j) = (N - j) r_A(i) + r_D,$$

and:

$$p_D(i, j) = (N - i) r_D(j) + r_A.$$

Finally the sample $p(i, j)$ is computed as the average of the two linear interpolations:

$$p(i, j) = \frac{p_A(i, j) + p_D(i, j)}{2},$$

where all predicted samples are approximated to the nearest integer.

Due to the fact that a relatively large number of samples is predicted using a small amount of information strongly localized, HEVC intra-prediction might still introduce unwanted prediction artefacts. In the case of angular prediction this is mostly evident when using modes with a strong directionality, for instance the pure horizontal mode: the entire block is predicted using exclusively the information in the samples immediately to the left (array A in Figure 2.20 (b)). The original samples in the block, particularly those in locations close to the right edge, might be very different from these predicted samples. This would provide considerably high residuals, localized in this area in the predicted block. These residual samples are difficult to compress, and an attempt to reduce the related bitrates might result in blocking artefacts in the decoded frames. Other types of intra-prediction (such as planar prediction) might also produce similar artefacts. To limit these effects particularly in large blocks, reference samples used for intra-prediction in HEVC can be filtered prior to prediction. HEVC makes use of a simple three-tap filter [91], which is applied only in particular intra-prediction modes and TU sizes. Filtering the reference samples prior to intra-prediction has the goal of distributing more smoothly the information in such samples consequently spreading the residual error more uniformly in the predicted block. This effect is illustrated in the example in Figure 2.21. Average absolute values of the residual samples obtained in the case of $16 \times 16$ blocks predicted using mode 12 in a test sequence encoded using HEVC is shown in case the smoothing filter is enabled (Figure 2.21 (a)), and in case it is disabled (Figure 2.21 (b)). In the second case, the residual sample magnitude clearly tends to increase towards the edges of the block, while a more uniform distribution of the residual magnitude is obtained when smoothing is enabled.

HEVC allows 35 possible modes for each PU of luma samples. To limit the bits needed to signal the choice of intra-prediction mode for the current PU, a list of 3 most-probable modes is considered using available information such as the optimal intra-prediction modes found for the PUs on top and left side of the current PU (if they are available).

If the intra-prediction mode chosen for the current PU is inside the list, an index is transmitted in the bitstream to signal which element in the list is used. Otherwise, the intra-prediction mode is fully signalled using a fixed codeword of 6 bits.

Chroma intra-prediction is performed after the prediction of luma. To limit complexity and also reduce the number of bits needed to signal the chroma intra-prediction mode, only up to 5 intra-prediction modes are allowed. Chroma intra-prediction is forced to make use of the same intra-prediction mode used to predict the luma component in case this is mode 0 (DC prediction), mode 1 (planar prediction), mode 10 (exactly horizontal angular prediction) or mode 26 (exactly vertical angular prediction). No additional information is transmitted in the bitstream in this case. Otherwise if the luma samples are predicted with a mode different than 0, 1, 10 or 26, these modes are all considered for chroma prediction along with a fifth mode (referred to as derived mode) that is set equal to the one used for the luma component. In this way, theoretically any of the 35 possible intra-prediction modes may be used also to predict chroma samples.

### 2.4.4 Inter-prediction in HEVC

A CU can be partitioned in PUs for inter-prediction in up to 8 different ways. Combined with the flexible CU size resulting from using the CTU structure, this allows the HEVC standard to define considerably more possible partitions for inter-prediction than AVC. In mode $2N \times 2N$ the full CU is predicted as a single PU of the same size. In modes

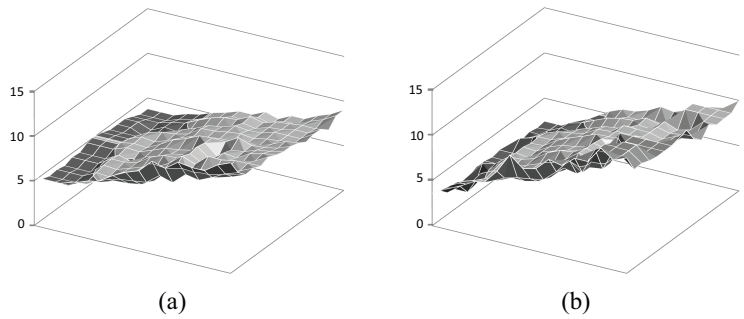



Figure 2.21: Example of average per-sample absolute residual magnitude. Smoothing is enabled in (a) and disabled in (b).

$2N \times N$ and $N \times 2N$, two PUs are independently predicted obtained by splitting the CU in two identical parts of half the height or half the width of the CU respectively. Mode $N \times N$ corresponds to partitioning the CU in four PUs of identical size. Finally HEVC also considers four additional inter-prediction modes referred to as Asymmetric Motion Partitions (AMP). The CU is split asymmetrically in two PUs. In modes $2N \times nU$ and $2N \times nD$ the CU is split in two PUs along the vertical side, where the height of the top-most PU is $\frac{3}{4}$ or $\frac{1}{4}$ of the CU height respectively. Similarly modes $nL \times 2N$ and $nR \times 2N$ are defined splitting the CU asymmetrically in the horizontal direction. Note that as opposite to intra-prediction where mode signalling happens at PU level but actual prediction happens at TU level, a single prediction is computed for the entire PU in inter-prediction. Also the partitioning used in HEVC inter-prediction is completely independent than the one used for transform, quantisation and entropy coding.

The inclusion of AMP and the large number of inter-prediction modes results in increased flexibility with respect to AVC, but comes at the cost of higher computational complexity especially at the encoder side if all possible modes are tested. To reduce complexity some modes can be disabled when testing CUs of particular sizes. For instance the $N \times N$ and AMP modes are typically not tested in $8 \times 8$ CUs. A graphical representation of the possible partitions of a CU for inter-modes is shown in Figure 2.22.

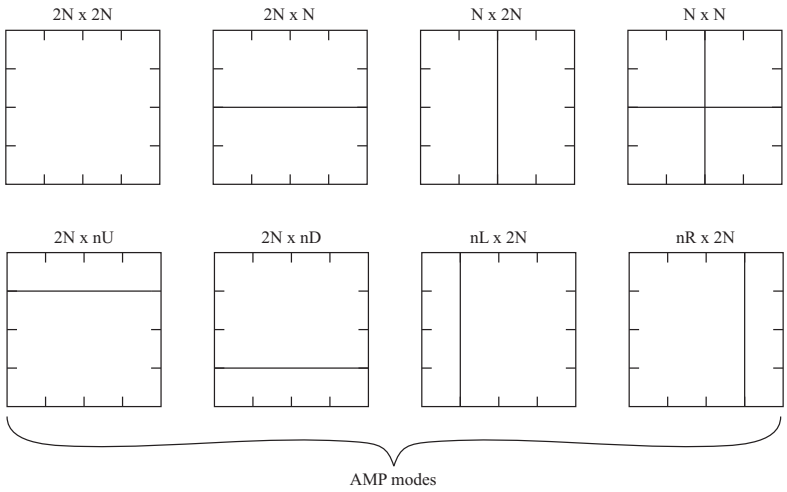



Figure 2.22: Inter-prediction modes in HEVC.

Two lists (list 0 and list 1) are used for the reference frames similarly to AVC. Also the way the lists are populated is almost identical to AVC. Again, CUs extracted from P slices are predicted using uni-directional motion estimation and motion compensation considering reference frames in list 0. Conversely CUs extracted from B slices make use of both lists and can be predicted using bidirectional inter-prediction.

The standard supports sub-pixel precision motion estimation up to quarter-precision accuracy. Especially when coding content at very high resolutions (such as UHD), it would be extremely expensive in terms of storage and resource allocation to keep track of all fractional samples in an interpolated reference frame. For this reason in HEVC interpolation is typically computed on-the-fly immediately before motion estimation or motion compensation only on the portion of reference frame currently needed for inter-prediction, drastically reducing the amount of resources needed to store the fractional samples. Clearly this requires the interpolation to be as fast as possible, and for this reason HEVC makes use of a simpler interpolation process than AVC, able to compute independently half-precision and quarter-precision samples [92]. Formally, denote as $p(m, n)$ the samples in the reference frame where integer values of $m$ and $n$ correspond to integer-precision samples. Half-precision samples are computed by means of an 8-tap filter with the following coefficients:

$$h = \frac{1}{64}\left[-1, 4, -11, 40, 40, -11, 4, -1\right].$$

Quarter-precision samples are computed by means of one of two 7-tap filters where one filter has coefficients:

$$q_1 = \frac{1}{64}\left[-1, 4, -10, 58, 17, -5, 1\right],$$

and the other filter $q_2$ has the same coefficients as $q_1$ in reverse order. $q_1$ or $q_2$ are selected based on the location of the nearest integer-precision sample to the current quarter-precision sample being interpolated. The interpolation process at either precision is

performed in two stages. In the first stage, fractional samples (both half-precision and quarter-precision) with one integer coordinate (either $m$ or $n$) are computed first using the available integer-precision samples. Assume for instance that the half-precision sample $p(m, n)$ is being interpolated where $m$ is integer. Integer-precision samples located in the horizontal direction in the same row as $p(m, n)$ are used for the interpolation. This is obtained as:

$$p(m,n) = \sum_{i=-3}^{4} \left(h(i)p(m, n - 0.5 + i)\right). \tag{2.6}$$

This is illustrated in Figure 2.23 for each of the three possible filters.

After the samples in the first stage have been computed these are interpolated using the same filters to compute the remaining samples.

A new tool is also considered in HEVC inter-prediction, referred to as Merge prediction, which includes and extends the functionalities of the SKIP and direct modes available in AVC. Unlike these modes that are always applied in AVC at a macroblock level, Merge prediction can be applied in HEVC either at a CU level (replacing the SKIP mode), or as a completely new technique at a PU level as illustrated here.

The idea of Merge prediction is again that of exploiting redundancy among motion vectors in neighbouring blocks to achieve higher compression. In fact it was observed [93] that the high granularity allowed by using the CTU structure results often in a number of neighbouring PUs sharing exactly the same motion information. By grouping together (i.e. merging) these PUs the motion information can be signalled only once per group hence reducing the related bitrates. In order to do so after a PU is inter-predicted its

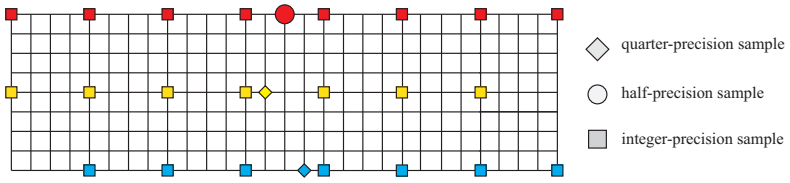



Figure 2.23: First stage of fractional interpolation in HEVC.

motion information is compared with its neighbouring blocks to check if there is a PU already coded that shares the same motion information. Only the parameters needed to point to that PU are transmitted for the current PU, instead of the entire motion information. In practice Merge prediction works by populating a list of maximum 5 candidates for each PU. The list might include spatial, temporal or virtual candidates. Spatial candidates are the motion information extracted from the 5 neighbouring PUs referred to as $M_{a1}$ to $M_{a5}$ in this order, as illustrated in Figure 2.24. Notice that these PUs can belong to the same CU as the current PU, or to a different CU in the same CTU, or to a different CUs in previously encoded CTUs. Each candidate (consisting of motion vector components and reference index) is only included if it is available and if it is not already in the list. Candidates may be unavailable for a variety of reasons: neighbouring blocks may not be inter-predicted, or they may be in a different slice, or the current PU may be adjacent to the boundaries of the frame. No more than 4 spatial candidates are included in the list. Up to two temporal candidates are then considered. These are the motion information $M_{b1}$ and $M_{b2}$ extracted in this order from two PUs in the previously encoded frame as illustrated in Figure 2.24. Notice that to avoid undesired decoding dependencies only the motion vectors are extracted from temporal candidates whereas the reference indexes are set to 0 (namely to point to the first element in the reference list). Finally, virtual candidates may be considered to fill the list in case there are available slots. First, the motion vectors and reference indexes of different spatial candidates (if more than two are available) are combined together to form new candidates. If there are still some free slots, zero-valued motion vectors with increasing reference indexes are included.

When Merge prediction is used as an alternative to the SKIP mode, the Merge candidate list is populated considering one single PU for the entire CU. A SKIP flag is signalled in the bitstream to enable the mode, along with an index to extract the correct candidate from the list. No other information is transmitted in the bitstream and the inter-predicted block is used as reconstruction similarly to the SKIP mode.

Conversely Merge prediction can also be used at a PU level during inter-prediction. After the motion information is computed for the PU, this is compared with the Merge candidates. In theory the method is designed to be used only in case exactly the same motion information is available among the Merge candidates. If this is the case, a Merge flag is set to true and signalled in the bitstream along with the merge index to select the candidate in the list. In practice, the encoder available in recent versions of the HM reference software allows the Merge candidates to be tested in an RD sense against conventional inter-prediction solution. A candidate at minimum RD cost may be selected to be used on a PU even if it is different than the actual optimal motion information derived for this PU. A flag is coded to signal if Merge prediction is used, and an index is possibly coded to signal the correct candidate. Note that when using Merge mode at a PU level, the rest of the encoding and decoding loop remain unchanged as if using conventional inter-prediction: residuals are computed, transformed, quantised and entropy coded. Merge mode was shown providing consistent compression improvements (around $-7\%$ BD-rates gains in standard conditions) with little impact on the coding complexity.

In case Merge prediction is not used on a PU, the motion information is signalled in the bitsteam, once for uni-directional predicted PUs or twice for bidirectional predicted PUs in B slices. Similarly to AVC only the motion vector difference is actually encoded

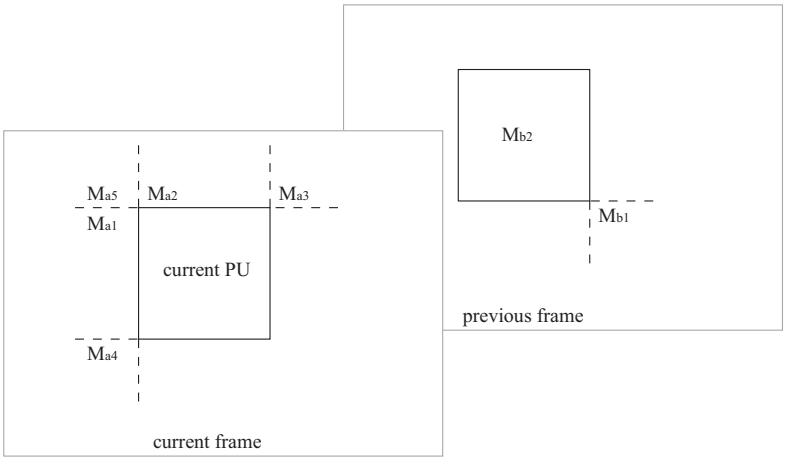



Figure 2.24: HEVC Merge candidates.

for each motion information, computed as the difference between the motion vector components and a motion vector prediction. This is derived in HEVC differently than in AVC, by means of a complex method referred to as Advanced Motion Vector Prediction (AMVP) [94]. The process of computing AMVP makes use of the same predictors used for populating the Merge candidate list (shown in Figure 2.24). Instead of considering a single predictor for each block as in AVC, some of these predictors are considered after the optimal inter-prediction solution is found, and the motion vector prediction is selected for a PU as the element in the list that is more similar to the optimal motion vector. A flag to identify this element is encoded in the motion information.

### 2.4.5 Transform and Other Tools in HEVC

In case a CU is not skipped, the residual samples are computed, transformed and quantised. It has been extensively proved that the decorrelating effects of DCT are mostly effective in case the transformed blocks contain relatively small amounts of texture changes. In order to capture and separate such changes, HEVC makes use of an additional level of partitioning where each CU can be divided in smaller square blocks referred to as transform units (TUs).

The partitioning of a CU in TUs follows an iterative approach referred to as Residual Quad-Tree (RQT) [95]. A TU of the same size of the CU is first considered. This is assigned a split flag to signal whether it is transformed as a whole or if it is split in four smaller TUs. In case it is split, each resulting TU is assigned a split flag and may be further partitioned in four, and so on. A minimum and maximum TU size are specified for the encoding, both of which within the allowed range of $32 \times 32$ to $4 \times 4$ samples. Also a minimum and maximum numbers of iterations of the RQT are specified in the encoder configuration. According to these parameters in certain conditions the split flags may be redundant and are not transmitted. For instance a TU is always split if it is larger than the maximum size (this is always true for $64 \times 64$ TUs) or if the level of recursion is lower than the minimum level; a TU is always not split if splitting would result in a

TU size smaller than the minimum size, or if the level of recursion is already equal to the maximum level.

Transform, quantisation and entropy coding are performed at a TU level. The transform base matrix used in HEVC was derived following a similar process as for the transform in AVC, by approximating to nearest integers DCT coefficients appropriately scaled [96]. To limit the resources needed while coding, a single transform base matrix $Q_{32}$ is defined for transforming the largest $32 \times 32$ TUs. Transform base matrices for smaller TUs are simply obtained by downsampling $Q_{32}$. For instance the matrix used for $8 \times 8$ TUs is [97]:

$$Q_8 = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix}.$$

The DCT has many desirable characteristics, but as already mentioned it was shown that it is not the optimal choice to decorrelate the residual signal especially in the case of intra-predicted blocks [52]. To better illustrate this behaviour consider for instance the case of a block of samples intra-predicted using exactly horizontal angular prediction. Samples located towards the right of the block are closer to the reference samples used for the prediction and therefore are likely to be predicted more accurately than samples closer to the left side of the block. Consequently the residual magnitudes can be expected to increase along with the distance of each sample from the reference array on the left. Conversely the DCT basis functions behave in the opposite way: for instance the function corresponding to the first frequency component ($n = 1$ in Figure 2.11) decreases monotonically. A much better representation of the signal would be obtained using a transform whose basis functions are more correlated with the general behaviour of the residual signal, such as the discrete sine transform (DST).

The DST is another member of the family of sinusoidal unitary transforms derived from discrete Fourier analysis which also includes the DCT. DST was originally proposed to be used in HEVC on all intra-predicted blocks. Later, a study on the compression performance provided by different transforms in the case of angular intra prediction was presented [53], showing that while more efficient compression is obtained using DST in intra-predicted blocks, the benefits of DST against DCT in large blocks are generally limited and do not counterbalance the disadvantages of its generically higher computational complexity and lack of fast algorithms. For this reason in the first version of HEVC DST is only used on small $4 \times 4$ TUs of luma samples [98].

Internal variables during the transform stages of HEVC are allowed a 16-bit representation, and such a limitation produces the need for truncation of variables. Consider as an example that a TU of $4 \times 4$ residual samples is being transformed using DCT, and assume an input 8-bit data representation. The dynamic range of the residual samples rises to 9 bits (to account for the sign, e.g. from $-255$ to $+255$). The first stage of transform consists in multiplying this block to the right by the transpose of the $4 \times 4$ DCT base matrix $Q_4$. The L1 norm of the transpose of $Q_4$ is $(64 \times 4 = 256)$, therefore the dynamic range of the variables after the first stage of transform goes from $-(256 \times 255)$ to $+(256 \times 255)$. This range would require 17 bits to be exactly represented. In order to keep variables within 16-bit representation, such variables must be scaled by a factor of 2 (i.e. 1-bit binary right shift). Extending this concept to blocks of arbitrary size $N \times N$ and input variables of arbitrary bitdepth $B$ (with $B > 8$), the variables after the first stage of transform must be shifted to the right by a number of bits equal to $s = \log_2(N) - 1 + (B - 8)$. The adjustments used in HEVC in the case of DCT for 8-bit input data representation are shown in Table 2-A for the two stages of transform.

After transform, coefficients are quantised and entropy coded. Quantisation in HEVC follows a very similar scheme as in its predecessor. Some details on the entropy coding methods used in HEVC can be found in Appendix A.

An interesting component of the HEVC coding scheme is represented by the final

filtering that is applied to a whole frame after the reconstruction of all CUs in the frame. First, a de-blocking filter is applied which is largely unchanged from the one in the H.264/AVC. This is not applied to $4 \times 4$ blocks of pixels at the boundaries in order to reduce the complexity. A new tool is also used called Sample Adaptive Offset [99]. The goal of the SAO is to reduce the distortion between the original frame and the reconstructed frames after de-quantisation and de-blocking filter. As such the SAO is not part of the inter-prediction (in fact it is applied to intra frames, and to intra blocks or after SKIP mode in inter frames as well), and only impacts the final PSNR of the current frame with no effects on the output of the prediction module. In particular, the encoder classifies the pixels in the reconstructed frame into different categories according to a set of rules to reduce the distortion, and thus extracts an optimal offset from a set of possible values in a look-up table. A trade-off between number of entries in the table and corresponding amount of side information needed to be transmitted is considered, and for this reason usually the number of categories is very small. The SAO parameters are calculated using the information available to the encoder after the current frame is encoded, and then are transmitted in the bitstream. At the decoder side, the relevant offset is applied to all the pixels in a frame belonging to the same category. Simulation results show that the SAO can achieve on average around 2% bitrate reduction and up to 6% bit rate reduction. As the additional complexity is very small, the runtime increases for encoders and decoders are only around 2%. An improved version of the SAO was proposed [100] which aims to reduce visual artifacts possibly introduced by the technique, and to reduce the buffer size required to store the parameters while encoding.

Table 2-A: Data ranges and adjustments during HEVC forward transforms with 8-bit input/output.

| TU Size | Max input | Input Bits | L1 norm | Max output | Output bits | Binary Shift |
|---|---|---|---|---|---|---|
| First DCT stage | | | | | | |
| $4 \times 4$ | +255 | 9 | 256 | +65280 | 17 | $\gg 1$ |
| $8 \times 8$ | +255 | 9 | 512 | +130560 | 18 | $\gg 2$ |
| $16 \times 16$ | +255 | 9 | 1024 | +261120 | 19 | $\gg 3$ |
| $32 \times 32$ | +255 | 9 | 2048 | +522240 | 20 | $\gg 4$ |
| Second DCT stage | | | | | | |
| $4 \times 4$ | +32767 | 16 | 256 | +8388352 | 24 | $\gg 8$ |
| $8 \times 8$ | +32767 | 16 | 512 | +16776704 | 25 | $\gg 9$ |
| $16 \times 16$ | +32767 | 16 | 1024 | +33553408 | 26 | $\gg 10$ |
| $32 \times 32$ | +32767 | 16 | 2048 | +67106816 | 27 | $\gg 11$ |

Finally note that also other tools were introduced in HEVC to further improve compression efficiency under different conditions, such as Transform Skip or Sign Data Hiding. While these are not of interest for the purpose of this thesis, a detailed description of all tools used in HEVC can be found in the literature.

# Chapter 3

# Low-complexity Adaptive Precision Motion Estimation

The proposed methods to decrease the computational complexity of motion estimation based on residual error characterisation and binary classification are presented in this chapter.

## 3.1 Adaptive Precision Motion Estimation

### 3.1.1 Previous works

Sub-pixel motion estimation is a fundamental component of video compression schemes and it is used in most standards since the very first attempts at video coding standardisation. The theoretical basis of the approach and practical implementations in AVC and HEVC were detailed in Chapter 2.

Due to its inherent high computational complexity, the role or sub-pixel motion estimation has been extensively investigated in the past. Already in 1993 a study was presented [39] with the goal of analysing the impact of fractional refinements on the

prediction accuracy of a typical video encoder under different coding conditions. The analysis showed that there exists a theoretical limit to the level of fractional accuracy after which any further refinement is likely to provide only minor improvements in terms of coding gains. For similar reasons even state-of-the-art and next generation video coding standards such as HEVC still only allow up to quarter-precision fractional accuracy. Recently it was shown [101] that finer motion vector refinements up to sixth-precision may be used to modestly improve compression efficiency, but only when used in a conditional way. Also interestingly, the study highlighted that there exists a correlation among the optimal level of fractional precision accuracy, the interpolation model, and the kind of content currently being considered. Sequences that contain high amount of motion typically resulted in better predictions when allowing higher levels of fractional precision accuracy, whereas half-precision accuracy may provide already sufficiently accurate predictions in the case of more static content.

Such idea was exploited in later works leading to the introduction of adaptive precision motion estimation [102]. Adaptive precision motion estimation consists of selecting the precision of the motion vector components while encoding, with the goal of providing high levels of accuracy and at the same time only performing sub-pixel motion estimation in a restricted number of cases. Such first attempts at adaptive precision motion estimation included two possible modes of execution. A first mode where the motion vector precision was estimated once for the whole frame and fixed throughout the frame, and an advanced mode where the fractional precision was estimated adaptively in a per-block basis, and could even differ for the horizontal and vertical motion vector components. The calculation of the optimal precision was obtained by measuring the impact of fractional components at different precisions on the final bitrate of the reconstructed frame. A simple encoding scheme was considered at this purpose in which residual samples are quantised directly in the spatial domain and immediately entropy coded to return a coding cost. While the approach is interesting in formalising a correlation between fractional precision accuracy and amount of texture in the frame,

unfortunately in modern video compression schemes the impact of the transform module in terms of final bitrate cannot be ignored and as such the idea is difficult to apply; similar works appeared later to extend and improve the approach [103] [104].

More recently another work was proposed [105] to address similar issues. When using such an approach, a parameter is computed for each block referred to as the deviation from flatness, depending on the residual error values obtained while performing integer-precision motion estimation. After the parameter is estimated for the currently encoded block, it is compared against a set of fixed thresholds to select the optimal fractional accuracy. The thresholds are computed by means of statistical analysis and are fixed throughout the encoding.

### 3.1.2 The need for adaptive precision

While in theory the global optimal solution at fractional precision should be derived following a full search at sub-pixel precision locations, in practice this kind of algorithms is very slow and rarely used. To reduce computational complexity most inter-prediction schemes make use of fast algorithms formed of a succession of steps. The outcome of the first step is a solution at integer-precision accuracy, which can be derived by means of full search or any fast algorithm as illustrated in previous chapters. The solution at integer-precision is then used as a starting point for the next step, to obtain a refined solution at half-precision accuracy. This is then used to obtain a refined quarter-precision accuracy solution and so on. Sub-pixel motion estimation algorithms based on successive fractional refinements are widely used in the context of modern video compression schemes and are used as the basis for the approach illustrated in this chapter.

Even when using fast algorithms, sub-pixel motion estimation still comes at the cost of higher computational complexity and also requires a larger number of bits to transmit the motion vector components. While this cost is often compensated in terms of better accuracy of the motion compensated prediction and consequently smaller residual

samples, in some cases the outcomes of integer-precision motion estimation would be already sufficiently accurate. In these cases the fractional refinements do not impact the motion estimation enough to compensate for the additional computational complexity, as illustrated in the following analysis.

In order to evaluate the impact of sub-pixel motion estimation refinements, some tests were performed on typical test sequences using the AVC standard. Test sequences have a resolution of $416 \times 240$ luma pixels, or are in the CIF format ($352 \times 288$ luma pixels), or in the QCIF format ($176 \times 144$ pixels). The list of tested sequences can be found in Table 3-A. Only I or P slices were considered in the experiments (namely no bi-directional prediction was considered), with the QP set to 22 for I slices and 23 for P slices. In all cases the full length of the sequences was considered. Intra-prediction modes were disabled while encoding P slices.

Some statistics were computed while encoding the test sequences. In particular, denote with $(m, n)$ the optimal motion vector found for a given block. In the following of this chapter the term block is used to refer to either a block or a sub-block, whichever is currently used for inter-prediction depending on the current mode being considered, regardless of its shape or size. Denote now a block as fractional if the optimal motion vector is fractional-valued, (namely $m$ or $n$ are not integers). In Table 3-A the percentage of fractional blocks is shown over the total number of inter-predicted blocks for each test sequence. Clearly, while always more than half of the blocks are predicted using fractional motion vector refinements, the percentage of fractional blocks varies considerably depending on the sequence. In some cases very few motion vectors are found at integer-precision, with as high as 89.6% fractional blocks in the case of the Racehorses sequence. In some other cases though the percentage of fractional blocks decreases drastically, with as low as to 12.5% of the total inter-predicted blocks in the case of the Akiyo sequence.

Although it can give a measure of how often sub-pixel motion estimation is used, counting the number of fractional blocks might not be sufficient to evaluate the actual impact of fractional solutions on the coding efficiency. Conventional encoders select

the optimal solution based on the computation of an RD cost. Fractional blocks are characterised by the fact that performing sub-pixel motion estimation returns an RD cost that is lower than that returned by integer-precision motion estimation, regardless of how large this gain is. At this purpose consider the difference between the optimal cost returned by sub-pixel motion estimation and the optimal cost previously found at integer-precision, and refer to this as the fractional difference. Such fractional difference can be normalised to the number of pixels in the block. Note that the larger is this fractional difference, the higher is the impact of the sub-pixel solution on the block, and consequently the higher is the expected coding gain as a consequence of using such a solution instead of the integer-precision solution.

The fractional difference was computed for all inter-predicted blocks in all the test sequences. Obviously all blocks that are not classified as fractional (i.e. whose motion vectors are integer valued) have a fractional difference equal to zero. The average value of the fractional difference was instead computed for all the other (fractional) blocks in all sequences. Refer to this average value as $\delta_{FD}$. When a fractional block returns a fractional difference higher than $\delta_{FD}$, the encoding is likely to benefit greatly from using sub-pixel motion estimation. Such blocks are referred to as fractional difference (FD) blocks. Conversely, all other blocks whose fractional difference is lower than $\delta_{FD}$ are likely to provide a sufficiently accurate solution even when using integer-precision motion estimation. Note that obviously FD blocks are a subset of fractional blocks. The

Table 3-A: Percentage of Fractional and FD Blocks over the total number of Inter-predicted Blocks

| **Resolution** | **Sequence** | **Fractional Blocks (%)** | **FD Blocks (%)** |
|---|---|---|---|
| $416 \times 240$ | Racehorses | 89.6% | 41.3% |
| | Keiba | 87.9% | 43.2% |
| | Mobisode2 | 44.8% | 4.4% |
| $352 \times 288$ | Crew | 88.8% | 28.9% |
| | Foreman (CIF) | 83.8% | 22.1% |
| | Soccer | 81.9% | 36.6% |
| | MotherAndDaughter | 45.4% | 9.1% |
| | Silent | 29.7% | 7.07% |
| $176 \times 144$ | Suzie | 70.2% | 21.3% |
| | Foreman (QCIF) | 49.1% | 21.9% |
| | Trevor | 47.8% | 19.8% |
| | Container | 19.9% | 1.4% |
| | Akiyo | 12.5% | 3.6% |

percentage of FD blocks over the total number of inter-predicted blocks in each test sequence is also shown in Table 3-A.

The results in Table 3-A highlight the fact that sub-pixel solutions are chosen in many cases even if the actual impact of using fractional refinements is relatively low. Consider the case of the Mobisode2 sequence. Fractional refinements are used on 44% of the inter-predicted blocks, at a considerably high cost in terms of computational complexity and encoding time. But sub-pixel motion estimation is capable of decreasing the RD cost by more than $\delta_{FD}$ only in a very limited number of cases (4.4%). Conversely, a fractional refinement is selected in 81.9% of the cases in the Soccer sequence, and almost half of these cases provide RD gains higher than $\delta_{FD}$ .

Some conclusions can be derived from this analysis. The number of blocks in which fractional refinements are actually effective varies considerably depending on the coding conditions and the currently encoded sequence. On the other hand the benefits of using sub-pixel motion estimation are too high to disable it completely, and such benefits have a big impact even in sequences in which fractional refinements are in fact used in a relatively small number of cases. Following from these observations, a novel method for low complexity motion estimation is proposed here which allows for consistent savings in computational time with a relatively low impact on the encoder performances, as shown in the following of this chapter.

### 3.1.3 Residual Error Surface Characterization

Due to the fact that typical motion estimation schemes derive solutions at different levels of precision in a succession of separate steps, it makes sense to exploit the solution found after the integer-precision step to make a decision regarding the following steps. In particular the distortion values computed when searching for the optimal integer-precision solution can be considered as a grid of sampled values extracted from a residual error surface. The method proposed in this thesis comes from the assumption

that the behaviour of this surface can be studied to predict whether the current integer-solution can be improved by means of fractional refinements. In case the residual error surface varies drastically in the surrounding of this solution, then moving away from the integer-solution may lead to a minimum of the surface whose corresponding distortion is considerably lower than the integer-solution distortion. This means that the fractional refinement can provide a gain sufficiently high to compensate for the additional computational complexity. Conversely, if the residual error surface is relatively flat then even if a minimum exists at a location with fractional coordinates, the distortion at this minimum is likely to be only slightly lower than the current distortion, hence providing only marginal gains. In these cases the integer-solution may be used to predict the current block with no significant losses in coding efficiency.

This idea requires the formalisation of a numerical metric to characterise the behaviour of the residual error surface. According to differential geometry, the curvature of a function $f(x)$ at a point $x_0$ can be defined as the second order derivative of the function evaluated at $x_0$. Extending this concept to functions of two independent variables $f(x, y)$, the curvature of a surface at a point can be defined in terms of the curvature of the functions obtained at the interceptions of the surface with normal planes that contain this point, at all possible directions. Consider a surface as shown in Figure 3.1. Consider a point $(x_0, y_0)$ in the surface, and denote as $L$ the normal vector to the surface at the point as shown in the figure. The vector $L$ can be obtained by taking the explicit form of the surface and then computing its gradient at $(x_0, y_0)$. Consider the normal planes to the surface at the point, i.e. the planes containing $L$, as illustrated in the figure. Finally consider the functions obtained as interception of the surface with such planes, and compute their curvature as the second order derivative of these functions evaluated at $(x_0, y_0)$. A common measure of the curvature of the three-dimensional surface is obtained from the maximum and minimum values of the curvatures of these intercepting functions. Such maximum and minimum curvature values are usually referred to as principal curvatures of the shape , denoted with $c_{min}$ and $c_{max}$. Similarly, the directions of the corresponding

normal planes are referred to as principal directions.

Depending on the principal curvatures, several measures of the curvature of a surface can be defined, including the Gaussian curvature (i.e. the product of the two principal curvatures) or the mean curvature (i.e. their arithmetic average). Another measure was proposed which is theoretically more suitable to capture the local shape of a surface than these other classical surface curvature metrics [106], referred to as curvedness and computed as:

$$C = \sqrt{{c_{min}}^2 + {c_{max}}^2}. \tag{3.1}$$

The curvedness was used in this work to measure the curvature of the residual error surface in order to characterise it for the purpose of selecting the fractional motion vector precision. In particular in the case of the residual error shape, only a sampled grid of values extracted from the surface is available, namely the distortion values at integer locations. In theory to derive the curvedness of this shape, full interpolation of such values could be computed and the curvedness could then be derived as from Equation 3.1. In practice this would be too expensive in terms of computational complexity. A

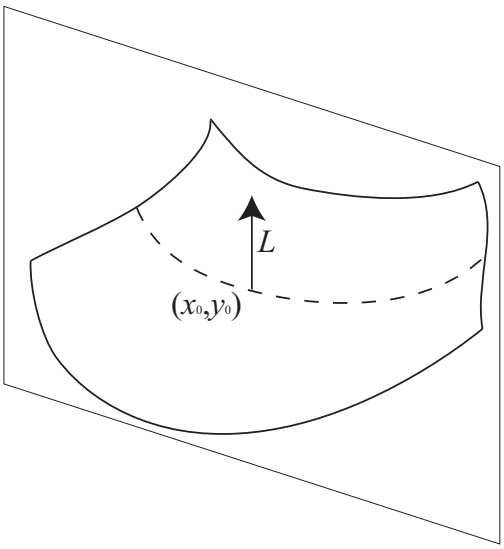



Figure 3.1: Curvature of a surface.

different technique is proposed in this thesis to obtain this value, referred to as sampled curvedness.

## 3.2 Sampled Curvedness

### 3.2.1 Computation of sampled curvedness

Assume that integer-precision motion estimation is performed in the current block making use of the SAD error metric. This could happen by means of full search motion estimation or any fast method. Notice that most fast motion estimation algorithms, such as for instance pattern-based or predictive zonal search algorithms, typically include a refinement step that involves computing a number of SAD values at several neighbouring locations before selecting the optimal solution. This means that after integer-precision motion estimation, the encoder has availability of the distortion values at several integer-precision displacements in the surrounding of the optimal solution even when using fast algorithms.

Denote as $x(i,j)$ the samples in the current block being encoded where $i = 0, ..., M-1$, $j = 0, ..., N-1$ and $M, N$ are the block height and width respectively. Denote also as $p(i+m, j+n)$ the samples in the currently considered reference frame at a certain displacement (namely the motion vector) $(m,n)$, again where $i = 0, ..., M-1$, $j = 0, ..., N-1$. The SAD between original block and reference block at displacement $(m,n)$ is defined as

$$SAD(m,n) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |p\,(i+m, j+n) - x\,(i,j)\,|. \tag{3.2}$$

Assume that the optimal motion vector (namely the output of integer-precision motion estimation) was found as $(m_0, n_0)$, and consider a window of $5 \times 5$ SAD values computed at locations $(m_0 + \triangle_m, n_0 + \triangle_n)$ where $\triangle_m, \triangle_n = -2, -1, ..., +2$. The

normal planes to the residual error surface at $(m_0, n_0)$ can be approximated as the 8 planes orthogonal to the $i, j$ coordinate plane and which contain $(m_0, n_0)$ and at least another point $(m_0 + \triangle_m, n_0 + \triangle_n)$, where either $\triangle_m \neq 0$ and/or $\triangle_n \neq 0$. Note that this is an approximation and in fact when using fast motion estimation algorithms the SAD at $(m_0, n_0)$ might not even be the minimum among the distortions at locations $(m_0 + \triangle_m, n_0 + \triangle_n)$. A representation of the 8 considered planes can be found in Figure 3.2 (a), denoted as (0) to (7).

The interception of each plane $i$ with the residual error surface can be approximated by means of polynomial interpolation as a function $d_i(t)$, where the continuous variable $t$ represents the distance from $(m_0, n_0)$. Each of these considered normal planes might intercept 3 or 5 distortion values at integer-precision locations, always including the distortion at the optimal integer-solution $SAD\,(m_0, n_0)$. For instance in the case of the planes (0) or (4) in Figure 3.2 (a), 5 integer-located SAD values are intercepted at values of $d$ equal to 0, $\pm 1$ and $\pm 2$. Similarly in the case of the planes (2) or (6) again 5 SAD values are intercepted, at values of $d$ equal to 0, $\pm\sqrt{2}$ and $\pm 2\sqrt{2}$. Finally in the case

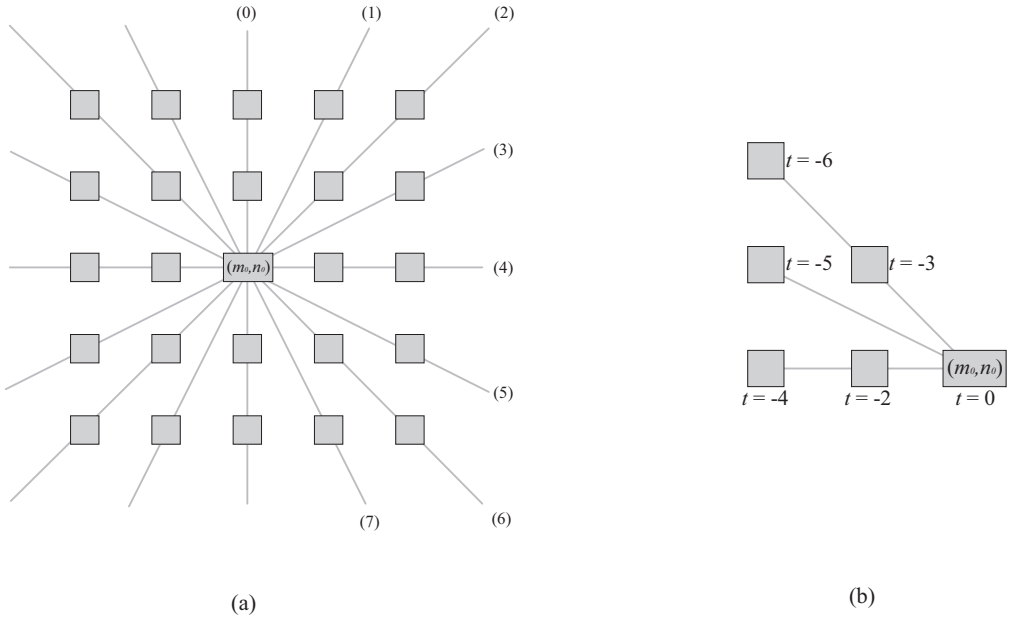



Figure 3.2: Normal planes intercepting integer-located SAD values (a). Approximated distance values (b).

of the remaining planes (1), (3), (5) and (7), 3 SAD values are intercepted at values of $d$ equal to 0 and $\pm\sqrt{5}$. In order to avoid square root operations which are typically computationally expensive, the distance values $d$ are approximated to integers as in the example in Figure 3.2 (b). In the case of a plane $(i)$ intercepting 3 distortion values (namely $i = 1, 3, 5, 7$), these are interpolated as:

$$d_i(t) = \frac{t(t-5)}{50} SAD\,(-1) - \frac{(t-5)(t+5)}{25} SAD\,(0) + \frac{t(t+5)}{50} SAD\,(+1)\,,$$

where $SAD\,(0) = SAD\,(m_0, n_0)$, and $SAD\,(\pm 1)$ are the two other SAD values intercepted by the currently considered plane $(i)$.

In the case of the planes (0) or (4) in Figure 3.2 (a) intercepting 5 distortion values, these are interpolated as:

$$\begin{aligned} d_i(t) = & \\ & = \frac{t(t-4)(t-2)(t+2)}{384} SAD\,(-2) - \frac{t(t-4)(t+4)(t-2)}{96} SAD\,(-1) + \\ & + \frac{(t-4)(t+4)(t-2)(t+2)}{64} SAD\,(-1) + \frac{t(t+4)(t-2)(t+2)}{384} SAD\,(+2) + \\ & - \frac{t(t-4)(t+4)(t+2)}{96} SAD\,(+1)\,, \end{aligned}$$

where again $SAD\,(0) = SAD\,(m_0, n_0)$, and $SAD\,(\pm 1)$, $SAD\,(\pm 2)$ are the four other SAD values intercepted by the currently considered plane. A similar expression can be found in the case of the planes (2) or (6) in Figure 3.2 (a), again intercepting 5 distortion values.

Double-differentiating $d_i(t)$ at $t = 0$, which corresponds to $(m_0, n_0)$, the following expressions are finally obtained:

$$c_i = \frac{SAD\,(-1)}{25} + \frac{SAD\,(+1)}{25} - \frac{2SAD\,(0)}{25}, \tag{3.3}$$

if $i = 1, 3, 5, 7$,

$$c_i = -\frac{SAD\,(-2)}{48} - \frac{SAD\,(+2)}{48} + \frac{SAD\,(-1)}{3} + \frac{SAD\,(+1)}{3} - \frac{5SAD\,(0)}{8}, \tag{3.4}$$

if $i = 0, 4$, and:

$$c_i = -\frac{SAD\,(-2)}{108} - \frac{SAD\,(+2)}{108} + \frac{4SAD\,(-1)}{27} + \frac{4SAD\,(+1)}{27} - \frac{5SAD\,(0)}{18}, \tag{3.5}$$

if $i = 2, 6$.

The maximum and minimum of the curvatures $c_i$ for $i = 0, ..., 7$, respectively denoted as $c_{max}$ and $c_{min}$, can be taken as an approximation of the principal curvatures of the residual error shape. These can then be used in Equation 3.1 to obtain an approximated curvedness, based on the available samples of the shape at integer-located values. To reduce complexity, the square root operation in Equation 3.1 is also avoided, and finally the following is obtained:

$$\hat{C} = c_{max}^2 + c_{min}^2, \tag{3.6}$$

where $\hat{C}$ is referred to as the sampled curvedness in the rest of this thesis. The sampled curvedness as defined here can be taken as an indication of how much the residual error surface varies in the surrounding of the integer-solution, and it can be easily obtained from the SAD values in the $5 \times 5$ window.

### 3.2.2 Sampled Curvedness and Fractional Motion Vector Precision

In order to validate the assumption that the behaviour of the residual error curvature on a block is correlated with the impact of sub-pixel motion estimation, the values of the sampled curvedness were studied in terms of coding efficiency at different levels of fractional precision. The same sequences shown in Table 3-A (under the same conditions) were used for this analysis. The sampled curvedness was computed for all inter-predicted

blocks in each sequence. Also, each inter-predicted block was classified as a FD block depending on its fractional difference in the same way as illustrated in subsection 3.1.2 (using the same value of $\delta_{FD}$).

The results of this analysis are shown in Table 3-B. The average sampled curvedness found for all FD blocks is presented in the table for each test sequence, along with the average sampled curvedness found for all other inter-predicted blocks. There is a clear correlation between the sampled curvedness values and the performance of sub-pixel motion estimation on each block. FD blocks, namely blocks in which fractional displacements provide very high gains in terms of RD cost, tend to assume higher values of the sampled curvedness than other blocks. The average sampled curvedness of FD blocks is higher than that of all other blocks in all test sequences but two, i.e. Silent (at CIF resolution) and Foreman (at QCIF resolution). In some cases the gap between the two average values is very high, for instance average sampled curvedness in FD blocks is more than four times that of all other blocks in the Mobisode2 sequence, and almost three times in the Container sequence. Conversely a small gap was found in some cases (for instance the Bowing sequence)

As an example, the values of the sampled curvedness found for the first 500 blocks of the Crew sequence are shown in Figure 3.3. FD blocks are represented with a circle whereas all other blocks are represented with a cross.

Table 3-B: Average sampled curvedness (SC) depending on optimal displacement.

| **Resolution** | **Sequence** | **Average SC of FD blocks)** | **Average SC of all other blocks** |
|---|---|---|---|
| $416 \times 240$ | Racehorses | 3.42 | 1.73 |
| | Keiba | 4.72 | 2.10 |
| | Mobisode2 | 2.92 | 0.70 |
| $352 \times 288$ | Crew | 2.80 | 1.33 |
| | Foreman (CIF) | 3.45 | 1.99 |
| | Soccer | 3.35 | 2.44 |
| | MotherAndDaughter | 2.15 | 1.59 |
| | Bowing | 2.54 | 2.53 |
| | Silent | 2.91 | 3.25 |
| $176 \times 144$ | Suzie | 2.87 | 1.22 |
| | Foreman (QCIF) | 4.44 | 4.82 |
| | Trevor | 3.95 | 3.08 |
| | Container | 16.21 | 6.08 |
| | Akiyo | 4.69 | 2.86 |

### 3.2.3 Binary Classification of Fractional Precision

The results in the previous subsection can be used to define a binary classifier, to select the precision of sub-pixel motion estimation on a per-block basis. Any time a block is inter-predicted, an integer-precision motion vector is first found as in conventional coding schemes. The grid of distortion errors at integer-precision locations is computed while searching for this motion vector. After the optimal solution is found, the sampled curvedness is computed using Equations 3.3 - 3.6. Then the following classifier is defined:

1. If the sampled curvedness of current block is below a fixed threshold $T$ no further action is required.

2. Otherwise the block is classified as a fractional block, and sub-pixel motion estimation is performed to find a fractional refinement for the motion vectors.

Next block is then considered and encoding continues as conventionally.

Finding a suitable value for the threshold $T$ is a crucial aspect of the proposed algorithm; using a large value of $T$ means that sub-pixel motion estimation is skipped in too many blocks, consequently decreasing the compression efficiency of the encoder.

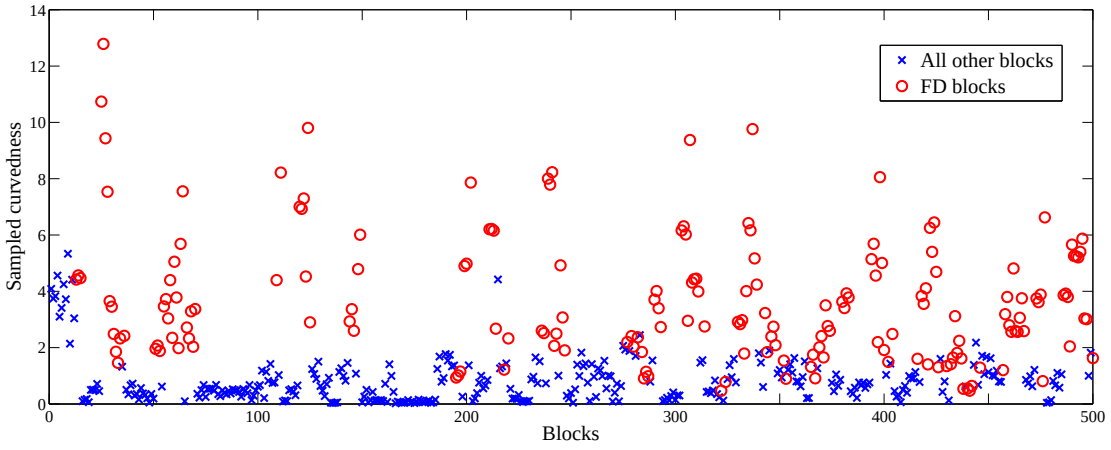



Figure 3.3: Sampled curvedness values for the first 500 blocks of the Crew sequence.

Conversely, using a small value of $T$ means that fractional refinements are computed in the majority of the blocks decreasing the algorithm performance. The optimal value of the threshold $T$ is such that the number of skipped blocks is maximased while keeping the losses in compression efficiency below an acceptable limit. At this purpose, the algorithm was tested on a number of test sequences using a variety of possible values of the threshold spanning from 0 (i.e.when no blocks are skipped) to 4, with a step of 0.4. Tests were then compared in terms of time savings and compression losses.

The performance of the algorithm in terms of compression losses can be measured by means of the BD-rate [19], a performance metric already defined in this thesis in subsection 2.1.4. At this purpose the method was compared with conventional AVC using four values of the QP (22, 27, 32 and 37, where QP is incremented by 1 in inter-predicted frames). For simplicity, a low-delay configuration consisting of an I frame followed by a fixed number of P frames is assumed here, but the approach can be easily extended to random-access configurations with no loss of generality.

The approach works by disabling sub-pixel motion estimation in a certain number of blocks, which means a theoretical maximum limit to the time that can be saved by the algorithm on the encoding of a sequence is given by the total encoding time obtained completely disabling sub-pixel motion estimation on all blocks. For this reason the gains of the algorithm in terms of time savings are measured with respect to such a theoretical limit. Formally denoting as $L_{subpel}$ the time saved by disabling completely sub-pixel motion estimation in seconds (obtained as the difference with respect to conventional encoding in terms of total encoding time at all tested QP values) and as $L_T$ the time saved by using the algorithm with a certain threshold $T$ (obtained in the same way), the performance of the algorithm is defined as $100 \times (L_T/L_{subpel})$ (in percentage).

Results of such tests can be found in Figure 3.4. The figure represents results obtained using the approach with all ten allowed values of the threshold (0.4 to 4 with a step of 0.4), encoding four sequences at CIF resolution. The vertical axis reports the BD-rate losses whereas the horizontal axis reports the time savings in percentage. Obviously, the

performances are greatly influenced by the value of the threshold $T$. Values smaller than 2 resulted in BD-rate losses no larger than 7% (in the case of the Soccer sequence) but at the advantage of at least 40% time savings. Larger values of the threshold could be acceptable for some sequences (less than 8% BD-rate losses are reported in the case of the Silent sequence even when using the largest threshold of 4) but they lead to high losses in other sequences.

While still able to provide beneficial results to the encoding (as reported in the rest of this chapter), the problem with this approach is that it cannot adapt to local characteristics specific to each sequence. Clearly the method performs very differently depending on the test sequence as shown in Figure 3.4. To better illustrate this problem consider for instance the case of the MotherAndDaughter sequence and assume the threshold was selected and fixed to $T = 2.4$. As reported in Table 3-B, while still FD blocks result in an average higher value of the sampled curvedness than other blocks, in general relatively low values of the sampled curvedness appear in the majority of the blocks throughout the sequence. A graphical representation of the sampled curvedness values found for a portion of the sequence (500 inter-predicted blocks) is presented in the plot in Figure 3.5 along with the fixed threshold. Clearly the algorithm as presented in this subsection

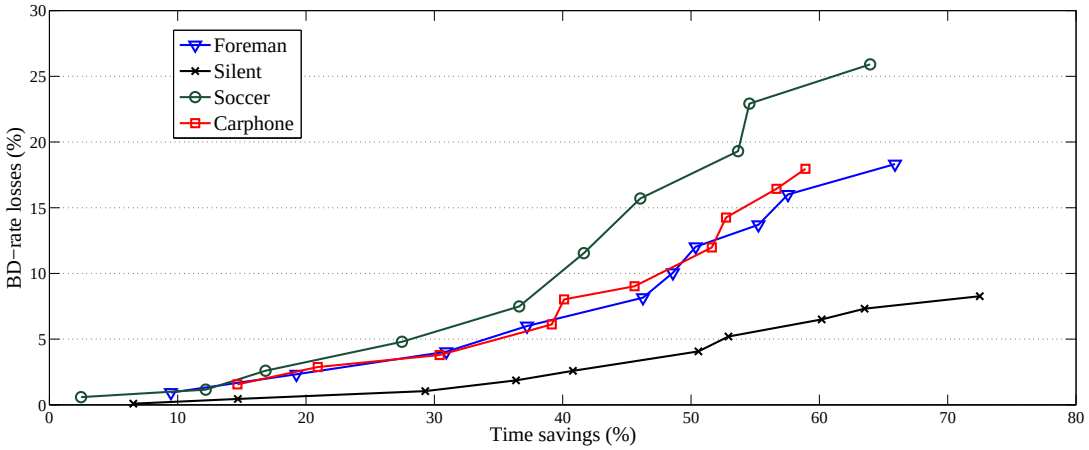



Figure 3.4: BD-rate versus time savings obtained using 10 threshold values (0.4 to 4 with a step of 0.4) when coding four sequences at CIF resolution.

is not optimal as a considerable number of FD blocks have a sampled curvedness value below the threshold. Sub-pixel motion estimation would be skipped in such blocks, with a corresponding decrease in the coding efficiency. In order to address these problems a different approach is proposed here as illustrated in the following section.

## 3.3 Adaptive Thresholding of the Sampled Curvedness

### 3.3.1 Initial estimate of Threshold

The approach proposed in the previous section entails two major issues. First, as already highlighted, global characteristics of each sequence are not taken into account. The sampled curvedness of all blocks is compared with a fixed predefined threshold whose computation is based on statistical analysis. Such threshold does not depend in any ways on the content being encoded. Second, local characteristics of the sequence are also not taken into account, regardless of the fact that these might lead to very high fluctuations of the sampled curvedness. For instance it is sensible to assume that large variations in the motion content of a sequence might result in similar oscillations in the average sampled curvedness values. Very high motion content usually correspond to high variations of the

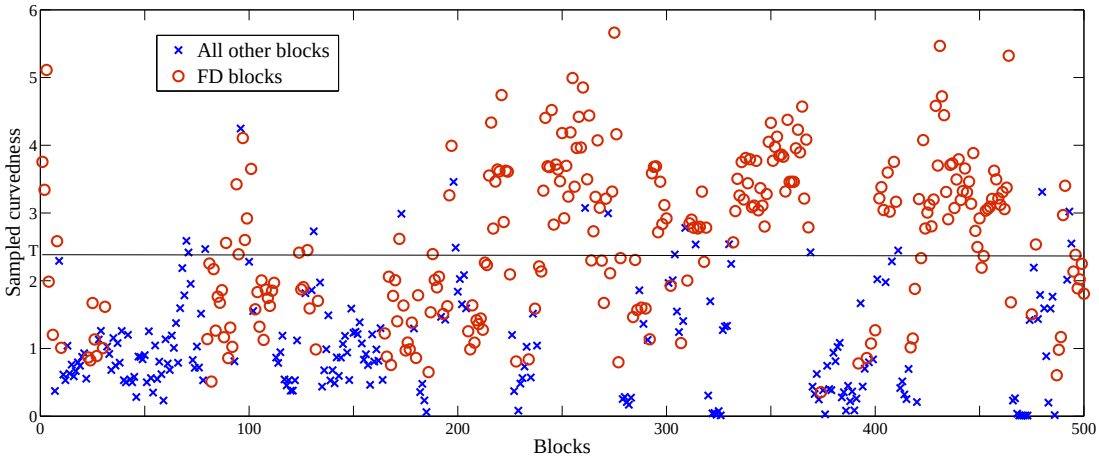



Figure 3.5: Sampled curvedness and fixed threshold for 500 blocks of the MotherAndADaughter sequence.

residual error as a result of even very small displacements, consequently providing high values of the sampled curvedness. On the contrary, portions of the sequence with very low motion content are likely to result in average lower values of the sampled curvedness. In such a situation using a fixed threshold would either imply that sub-pixel motion estimation is skipped in too many blocks (in portions of the sequence containing low motion) or it is performed too many times (in portions containing high motion).

Figure 3.4 shows that the relationship between a certain threshold $T$ and the corresponding performance losses is highly dependent on the content being encoded. While the actual effects of using this threshold in terms of bitrate losses are difficult to compute, an estimate of such losses can be obtained by computing the sum of all fractional differences (namely the difference between distortion obtained using and not using sub-pixel motion estimation respectively) which are "lost" as a result of using a certain threshold $T$. Such quantity can be used to predict the effects of a threshold in terms of actual performance losses, and it is referred to here as the fractional impact.

Following from this definition, a different approach is presented in this section where the threshold is initially estimated using a group of training blocks, and then sequentially updated to adapt to local characteristics of each sequence depending on the fractional impacts. The goal of the approach is that of selecting a predetermined target fractional impact $E_{target}$, and then finding a corresponding value of the threshold $T_{target}$ to meet such a target. Both aforementioned issues are taken into account. First, characteristics of each sequence are taken into account by computing an initial estimate of $T_{target}$ using information obtained coding a number of blocks at the beginning of the sequence. Then, local variations within the sequence are also taken into account by successively updating the threshold at fixed intervals, to make sure that the fractional impact produced while encoding is as close as possible to the target value $E_{target}$.

At this purpose, assume that an AVC encoder is used to code a certain sequence. The encoder partitions each frame in macroblocks, then tests a number of inter-prediction modes on each macroblock possibly further partitioning each macroblock in a number

of blocks and computing motion estimation on each block. Refer to each inter-predicted block using an index $k$, where $k = 0, 1, ....$ Blocks are assigned incremental values of $k$ in the same order as they are considered by the encoder. Also, in case the encoder is testing inter-prediction on the same portion of a frame several times (for instance when considering different modes on a macroblock), these are considered as individual blocks and separately assigned different indexes.

Consider now the group of inter-predicted blocks $k = 0, 1, ..., K-1$, namely the first $K$ blocks considered by the encoder when coding this sequence. In case an I frame is encountered which does not require a decoder refresh, the blocks within this frame are not considered while filling the group; this is completed with inter-predicted blocks extracted from the next P or B frame. If a decoder refresh is required, the encoding starts over from $k = 0$ as if at the beginning of a new sequence.

Consider now a set of thresholds $T_i$ where $i = 0, 1, 2, ..., L-1$ and $L$ is a predefined integer parameter. The values should be chosen such that $T_i < T_j$ if $i < j$. The fractional impacts obtained as an effect of using each threshold $T_i$ while coding the $K$ blocks in the group can be easily obtained by considering a corresponding parameter $E_i$. At the beginning of the encoding of the sequence these are set to zero, or $E_i = 0$ for $i = 0, 1, 2, ..., L-1$.

The encoder starts coding each block $k$ as in conventional AVC. After performing integer-precision motion estimation, Equations 3.3-3.6 are used to compute the sampled curvedness of the block, referred to as $\hat{C}_k$. Then sub-pixel motion estimation is performed. The fractional difference of the block $D_k$ is also computed as the difference between optimal cost output by sub-pixel motion estimation and the cost previously output by integer-precision motion estimation. Finally the fractional impacts of each threshold value are updated as follows.

For each $i = 0, 1, 2, ..., L-1$, if:

$$\hat{C}_k < T_i,$$

then:

$$E_i = E_i + D_k.$$

This is repeated for all blocks until $k = K - 1$. After $K$ blocks are encoded, the obtained values $E_i$ represent the fractional impact as an effect of using each threshold $T_i$ on such blocks. It is easy to verify that $T_i \leq T_j$ if $i < j$.

Assume now that a certain target fractional impact $E_{target}$ is considered. This can be selected prior to the encoding based on a trade-off between time savings and losses in compression efficiency, where larger values of $E_{target}$ correspond to higher losses in compression efficiency. The values of $T_i$ and corresponding $E_i$ can be fitted to define a complete function between $T$ and $E$. While more complex models could be used, for simplicity and in order to limit the complexity of the approach, a piecewise linear function is assumed here as illustrated in the example in the plot in Figure 3.6 for $L = 4$. By appropriately selecting the number $L$ and the values of $T_i$, this function can provide a sufficiently accurate representation of the behaviour between $T$ and $E$.

Finally the value of the target threshold $T_{target}$ can be computed as follows:

1. If $E_1 \leq E_{target}$, $T_{target} = (T_1 - T_0)\frac{E_{target} - E_0}{E_1 - E_0}$.
2. Otherwise if $E_i < E_{target} \leq E_{i+1}$ where $1 \leq i \leq L - 3$, $T_{target} = (T_{i+1} - T_i)\frac{E_{target} - E_i}{E_{i+1} - E_i}$.
3. Otherwise $T_{target} = (T_{L-1} - T_{L-2})\frac{E_{target} - E_{L-2}}{E_{L-1} - E_{L-2}}$.

A threshold $T = T_{target}$ selected with the aforementioned method approximately

corresponds to the threshold that produces a fractional impact equal to $E_{target}$ while coding the initial group of $K$ training blocks. Under the assumption that the following blocks behave similarly to the training blocks, setting the threshold to $T = T_{target}$ should ensure that roughly the same fractional impact is obtained after coding the next $K$ blocks and so on. As a result the proposed algorithm should be able to approximately predict the expected losses in compression efficiency, and also to level such losses to a predefined limit which can be controlled by appropriately setting the value of $E_{target}$

### 3.3.2 On-the-fly Update of the Threshold

Once the initial value of the threshold is estimated on the training blocks by means of the method in the previous subsection, the encoder inputs all following blocks to the algorithm in subsection 3.2.3. The sampled curvedness is computed for each inter-predicted block after integer-precision motion estimation. If this is larger than $T$ the block is coded as in conventional schemes, otherwise if it is below $T$ sub-pixel motion estimation is skipped, hence reducing complexity.

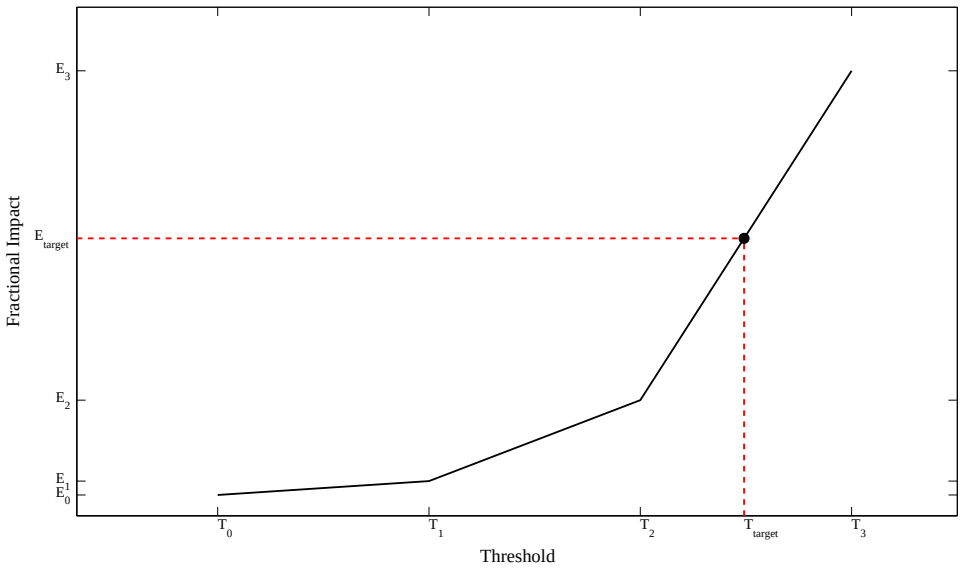



Figure 3.6: Selection of the target threshold based on fractional impacts.

The problem with such a method is that this value of the threshold might not be optimal when coding the following blocks. This can happen due to several factors. First, the behaviour of the first $K$ blocks might not be sufficiently indicative to model the relationship between $E$ and $T$ in the rest of the sequence. Second, such a relationship might simply change throughout the sequence, which means a fractional impact very different than the expected value $E_{target}$ could be obtained when coding blocks distant in time from the first group. Finally, the sampled curvedness might not be sufficiently correlated with the outcomes of motion estimation which means a generally low performance of the approach could be expected in case sub-pixel motion estimation is skipped in many blocks.

In all these cases the proposed approach as presented so far would fail, possibly providing high compression losses. To prevent this behaviour the encoder must ensure that the selected threshold $T_{target}$ is still a good approximation to provide a fractional impact approximately equal to $E_{target}$ when coding the current portion of the signal, namely that the relationship between $E$ and $T$ in this portion is not drastically different than the model estimated on the training blocks as in Figure 3.6.

Unfortunately, a complete model of such a relationship cannot be defined when coding blocks input to the algorithm in 3.2.3. All blocks whose sampled curvedness fall below the current threshold $T$ are not input to sub-pixel motion estimation, which means the corresponding fractional difference cannot be computed. In other words once a block is classified as not fractional, it is clearly not possible to compute the impact of fractional refinements on predicting its content. Therefore the fractional impact due to using $T_{target}$ on such blocks is unknown, as represented by the grey area in the left of the plot on top of Figure 3.7.

In order to compute a complete model of the relationship between $T$ and $E$, the encoder has no other option than to consider another group of $K$ blocks and code these blocks with full sub-pixel motion estimation, and then calculate a new value of $T_{target}$ based on the outcomes of such blocks.

On the other hand, computing the threshold too often corresponds to drastically limiting the benefits of the proposed method: each time the threshold is recomputed, $K$ blocks are coded without using adaptive precision motion estimation. For this reason the encoder should be able to selectively decide whether the current threshold is still good enough to encode the current portion of the sequence, or if the threshold should be recomputed. A method to perform such a decision is shown here.

Consider the group of $K$ inter-predicted blocks $k = K, K+1, ..., 2K-1$, namely the first $K$ blocks coded after the training blocks. These are input to the proposed algorithm using the previously calculated $T = T_{target}$. Now consider again the values

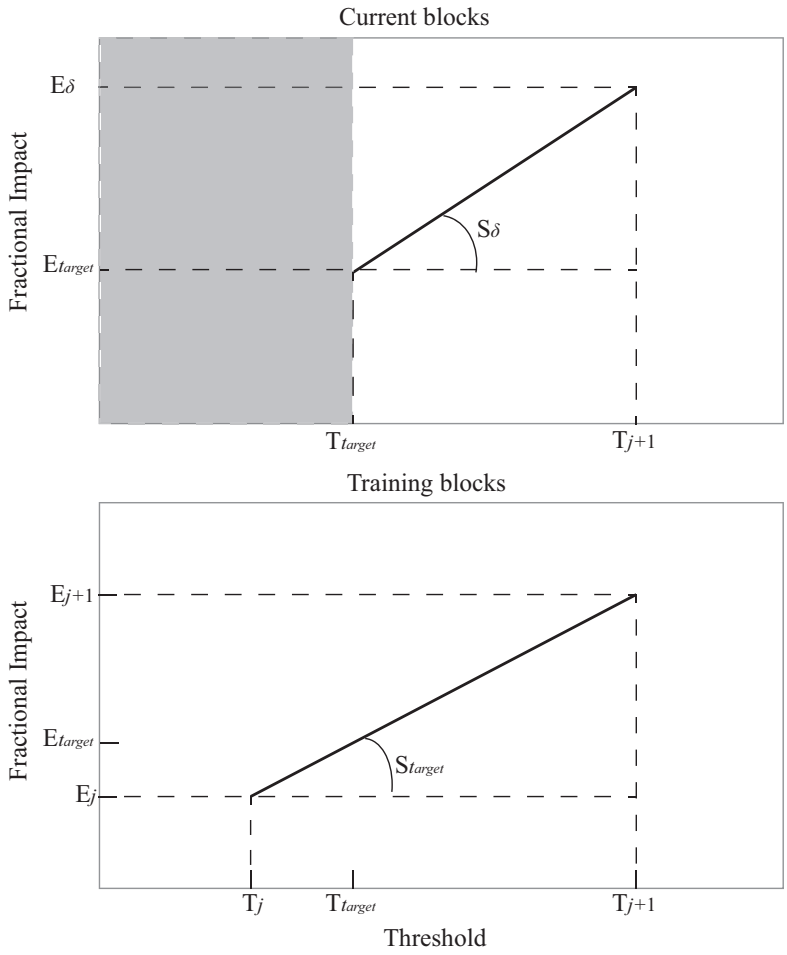



Figure 3.7: Slope of fractional impact as a function of threshold for current and training blocks with a threshold $T_{target}$.

$T_i$ where $i = 0, 1, 2, ..., L-1$, used when estimating the initial threshold to model the function between $E$ and $T$, and assume that $T$ was selected such that $T_j < T \leq T_{j+1}$. Denote as $\delta = T_{j+1} - T$ the difference between the current threshold and the smaller $T_i$ larger than $T$. The delta fractional impact $E_\delta$ between $T$ and $T + \delta$ in this group of blocks can be obtained by summing all fractional differences of all blocks in the group whose sampled curvedness is such that $T < \hat{C}_k \leq T+\delta$. Finally the slope $E$ as a function of $T$ in the close proximity of $T_{target}$ can be computed as $S_\delta = E_\delta/\delta$, as illustrated in the top of Figure 3.7. Also, define as $N_K$ the number of blocks in the group such that $\hat{C}_k < T$, namely blocks in which fractional refinements are not computed.

Consider now the slope of the piecewise linear function used for the training blocks computed at $T = T_{target}$. This can be calculated as $S_{target} = (E_{j+1} - E_j)/(T_{j+1} - T_j)$ where again $j$ is such that $T_j < T_{target} \leq T_{j+1}$, as illustrated in the bottom of Figure 3.7. Also, denote as $N_0$ the number of training blocks $k = 0, 1, ..., K-1$ such that $\hat{C}_k < T$.

The parameters $S_\delta$, $S_{target}$, $N_K$ and $N_0$ can be used to determine if the relationship between $E$ and $T$ diverge drastically from the model used to estimate $T_{target}$ by defining the following two conditions:

1. The number $N_K$ is very different than $N_0$, namely fractional refinements are skipped a larger (or smaller) number of blocks in the current group than in the initial group. By appropriately selecting a maximum distance $\Delta_N$ this can be formalised as: $|N_K - N_0| > \Delta_N$.

2. The slope $S_\delta$ is very different than $S_{target}$. By appropriately selecting a maximum distance $\Delta_S$ this can be formalised as: $|S_\delta - S_{target}| > \Delta_S$.

Condition 1 identifies the case when a larger (or smaller) number of blocks than in the training blocks result in a value of the sampled curvedness below the threshold. Correspondingly a larger (or smaller) fractional impact can be expected, which consequently means that the current threshold does not represent accurately such blocks. Condition 2 corresponds to cases when the predicted relationship between $T$ and $E$ in the current

group is different than the model used to select the threshold. The assumption that $E_{target}$ would be obtained using $T_{target}$ depends on such a model, therefore a new model must be defined.

In case either of these conditions is satisfied after coding the current group of blocks, the index $k$ is reset to 0, and the next $K$ blocks are used for training the classifier, to compute a new threshold as if at the beginning of a new sequence. Otherwise a new group is considered using the same value of $T$. The parameters $N_K$ and $S_{target}$ are computed for each group and Conditions 1 and 2 are used to decide whether a new threshold should be computed for the next group.

The proposed method is capable of obtaining considerable time savings while at the same time limiting the performance losses, as shown in the next section.

## 3.4 Results

The methods proposed in this chapter were implemented and tested in the context of the AVC reference software version JM 18.5 [72]. A total of 14 sequences were tested at different resolutions. These are common test sequences used in the literature, or sequences used by the JVT or JCT-VC expert groups while developing AVC and HEVC respectively. All test sequences are 300 frames long at a framerate of 30 fps. In particular the following sequences were used:

1. Resolution of $416 \times 240$: Mobisode2, RaceHorses, Keiba.
2. Resolution of $352 \times 288$: Bowing, Carphone, Crew, Foreman, Silent, Soccer.
3. Resolution of $176 \times 144$: Akiyo, MotherAndDaughter (also abbreviated as MotherAndD.), News, Suzie, Trevor.

Due to the fact that most of the experiments used to derive parameters needed by the proposed methods were performed under a low-delay configuration, this configuration is

also assumed in all tests in this section. The implementation and results could be easily extended to other configurations but the parameters used throughout the methods might require some adaptation.

The proposed approaches work by selectively disabling sub-pixel motion estimation on a number of blocks, and for this reason test sequences were encoded to show performance of AVC when sub-pixel motion estimation is enabled or completely disabled respectively. Results of these tests are shown in Table 3-C. Compression efficiency is measured in terms of the BD-rate (in percentage) where conventional AVC with enabled sub-pixel motion estimation as implemented in the JM reference software is used as anchor. Time savings are shown in terms of number of seconds saved disabling sub-pixel motion estimation with respect to full AVC. QP values of 22, 27, 32 and 37 (where QP is incremented by 1 in P frames) were used. All other configuration parameters are identical between the two encoders.

Notice that the encoding times presented in the table are affected by the configuration parameters or the resolution of the sequence being coded. It is reasonable to expect very different results as an effect of changing the integer-precision motion estimation search window or search algorithm. Nonetheless, the time difference obtained when completely disabling sub-pixel motion estimation can be taken as a a maximum limit of the time savings which can be obtained by the proposed approaches. For this reason in order

Table 3-C: Performance of conventional AVC enabling and disabling sub-pixel motion estimation.

| **Sequence** | **Time(no sub-pel)** | **Time (with sub-pel)** | **Time Difference** | **BD-rates (%)** |
|---|---|---|---|---|
| Mobisode2 | 58.5 | 108.3 | 49.8 | 14.74 |
| RaceHorses | 90.8 | 188.0 | 97.3 | 39.70 |
| Keiba | 86.9 | 170.3 | 83.5 | 23.70 |
| Bowing | 64.1 | 108.4 | 44.3 | 14.67 |
| Carphone | 66.4 | 133.9 | 67.4 | 35.71 |
| Crew | 77.2 | 158.6 | 81.4 | 42.74 |
| Foreman | 70.5 | 138.1 | 67.6 | 35.01 |
| Silent | 62.6 | 108.3 | 45.7 | 19.01 |
| Soccer | 76.8 | 144.8 | 67.9 | 34.72 |
| Akiyo | 16.4 | 25.0 | 8.6 | 16.92 |
| MotherAndD. | 15.9 | 26.3 | 10.3 | 44.00 |
| News | 17.7 | 28.6 | 11.0 | 23.41 |
| Suzie | 16.8 | 34.6 | 17.8 | 41.07 |
| Trevor | 18.3 | 33.6 | 15.3 | 35.78 |

to present results in a way as independent as possible from the coding conditions, such time savings are measured with respect to this time difference. Denoting as $L_{subpel}$ the time saved by disabling completely sub-pixel motion estimation (namely the values in the fourth column in Table 3-C) and as $L$ the time saved by using the currently tested approach, time savings are computed as $\Delta Time = 100 \times (L/L_{subpel})$ (in percentage).

First, the algorithm in subsection 3.2.3 making use of a fixed threshold value was tested. The results of such tests were compared with the previous state-of-the-art method for adaptive precision motion estimation based on the deviation from flatness [105], referred to as DF in the rest of this section. In order to show full performance of the approach, the proposed algorithm was tested using a variety of threshold values spanning from 0.4 to 5.2, with a step of 0.4.

Full results are presented in Table 3-D. Performance of the approach in terms of BD-rates and time savings (denoted as $\Delta Time$ in the tables) using all tested values of the threshold are reported, compared with results obtained using the DF method (shown in italic in the table). BD-rate results are always presented in percentage, where positive values represent losses in compression efficiency with respect to the anchor. Results in bold correspond to sequences in which larger time savings are obtained using the proposed approach than DF, under the condition that performance losses are smaller or almost the same (not more than 1% higher) than those obtained using DF.

Clearly very high time savings can be obtained using the approach, with as high as 70% time savings in some sequences. On the other hand, performance of the method with respect to losses in compression efficiency greatly depends on the threshold value. Considerable time savings at the cost of acceptable losses are reported in some sequences, whereas very high losses are reported in other cases with losses higher than 20% in terms of BD-rates in some test sequences. As a general behaviour, thresholds between 0.8 and 2.0 provided best results; values of the threshold within this range resulted in better performance of the proposed approach than DF in 12 out of 14 test sequences.

The results in Table 3-D highlight that both the proposed method using fixed threshold, and previously proposed adaptive precision motion estimation methods such as DF, suffer from the fact that performances are highly influenced by the current coding conditions. Time savings and BD-rate losses grow with the increasing values of the thresholds, but no value of the threshold can be selected which allows for consistent time savings in all sequences, without affecting too much the coding efficiency in some of the sequences. In particular sequences such as Crew and Soccer result in high performance losses already with thresholds higher than 0.8. Conversely much higher thresholds could be used on other sequences (with correspondingly higher time savings) with acceptable BD-rate losses.

The approaches proposed in Section 3.3 were proposed to solve these issues and are tested here. The number of training blocks $K$ used to compute $T_{target}$ was set to 2000. Tests are performed using a variety of different target fractional impacts $E_{target}$. Assuming that the fractional difference of each block $D_k$ in Subsection 3.3.1 is normalised to the block size, and assuming that the final fractional impacts $E_i$ are normalised by the number of blocks $K$, then the target fractional impact $E_{target}$ represents the target cost loss per pixel expected as an effect of using the proposed method. To analyse the performance of the approach under different conditions, values of $E_{target}$ from 1 to 8 with a step of 1 were tested.

First, some tests were performed in case the threshold is initially estimated but is then fixed for the entire length of the sequence. This was tested varying the number of encoded frames, to show that BD-rate losses tend to increase as an effect of increasing the number of frames in the sequence. A selection of these results is reported in Table 3-E for 6 sequences (2 for each tested resolution), encoding 10, 20, 50 frames and the full length of each sequence.

Results in the table highlight the fact that when the threshold is fixed per sequence, performances of the method decrease with the number of frames encoded in the sequence. Limited losses in compression efficiency were reported when coding only 10 frames even

for relatively high values of $E_{target}$. Conversely, unacceptable losses are reported for almost all values of $E_{target}$ when coding full length of the sequences.

Finally, results of the approach when the threshold is updated on-the-fly throughout the encoding are presented. Full results on all test sequences are reported in Table 3-F. Full length of the sequence was used in these tests. For comparison purposes the table also reports results obtained using fixed threshold. This is shown for a single value of the threshold T=1.6, which is the threshold that provided best results as compared with the previous state-of-the-art DF approach [105] in terms of number of test sequences in which it performed better than DF (as shown in Table 3-D).

The major achievement of the approach is its ability to maintain an almost constant compression efficiency regardless of the encoded sequence or local variations in the signal. Also, the approach provides a reliable way of selecting an expected amount of performance losses by appropriately selecting a desired target fractional impact value $E_{target}$ prior to encoding. Each increasing value of the target fractional impact resulted in successively higher compression efficiency losses in all test sequences. The mean and variance of the BD-rate losses found for all test sequences, for each value of the target fractional impact as well as for the fixed threshold approach, are also shown in Table 3-F. The fixed threshold approach resulted in a variance of the BD-rate losses more than 6 times higher than the maximum variance found using the adaptive threshold method, for an almost identical value of the mean. Sequences with high average fractional impacts (such as Crew or Soccer) resulted in considerably smaller losses, whereas sequences in which sub-pixel motion estimation is shown to have a smaller impact (such as News or Akiyo) were allowed higher losses and correspondingly bigger time savings.

In terms of time savings, the method is shown consistently reducing the time as compared with conventional AVC. Clearly the approach requires itself some additional computational complexity, in order to calculate the sampled curvedness for each inter-predicted block, and also to keep track of the variables needed to update the thresholds and to perform the threshold computation when needed. Nonetheless this complexity is

always compensated by the time saved as a result of avoiding fractional refinements in a selection of blocks. Even when using the smallest thresholds (which means sub-pixel motion estimation is avoided in the smallest amount of blocks), still considerable time savings were obtained.

As a complement to such results, the RD curves showing PSNR versus bitrate obtained for the four considered values of the QP when using $E_{target} = 4$ are shown in Figure 3.8, 3.8 and 3.8 for the Foreman, Crew and MotherAndDaughter sequences respectively. Curves are shown for the proposed approach compared with original AVC and AVC without sub-pixel motion estimation. From the plots it is evident that RD performance of the proposed approach is very similar to conventional AVC, with very small losses in compression efficiency as opposite to completely disabling sub-pixel motion estimation. A visual comparison of two decoded frames extracted from the Crew and Foreman sequences respectively under the same conditions is shown in Figure 3.9. With some effort, some small additional artefacts can be noticed in the frames produced by the proposed approach (for instance in the collar of the man in the Foreman sequence). Such artefacts are very limited and do not affect the overall quality of the frames. Similar effects were observed in other frames and in other test sequences.

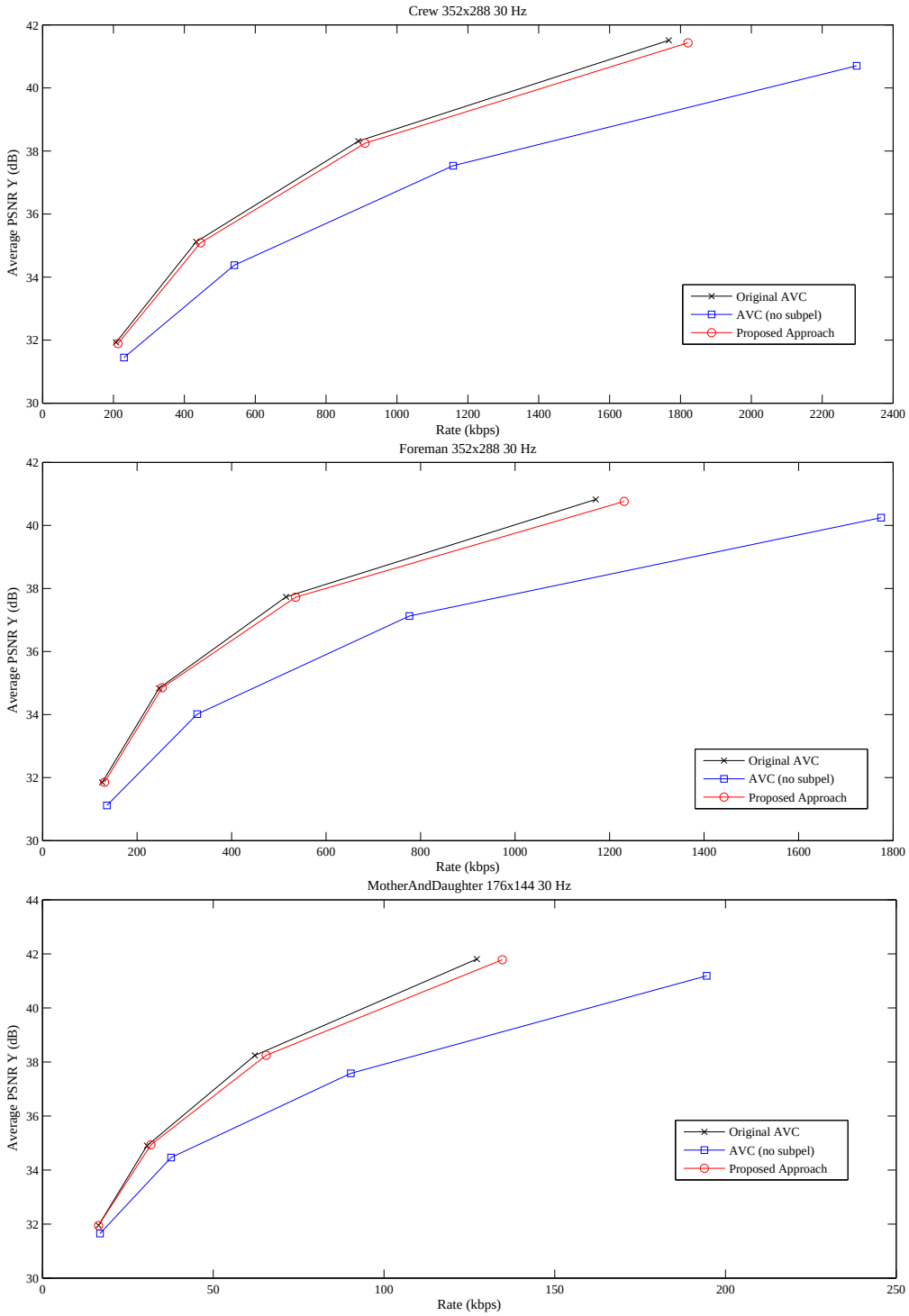



Figure 3.8: Rate-distortion curve of the proposed approach using adaptive threshold compared with conventional AVC and AVC with disabled sub-pixel motion estimation.

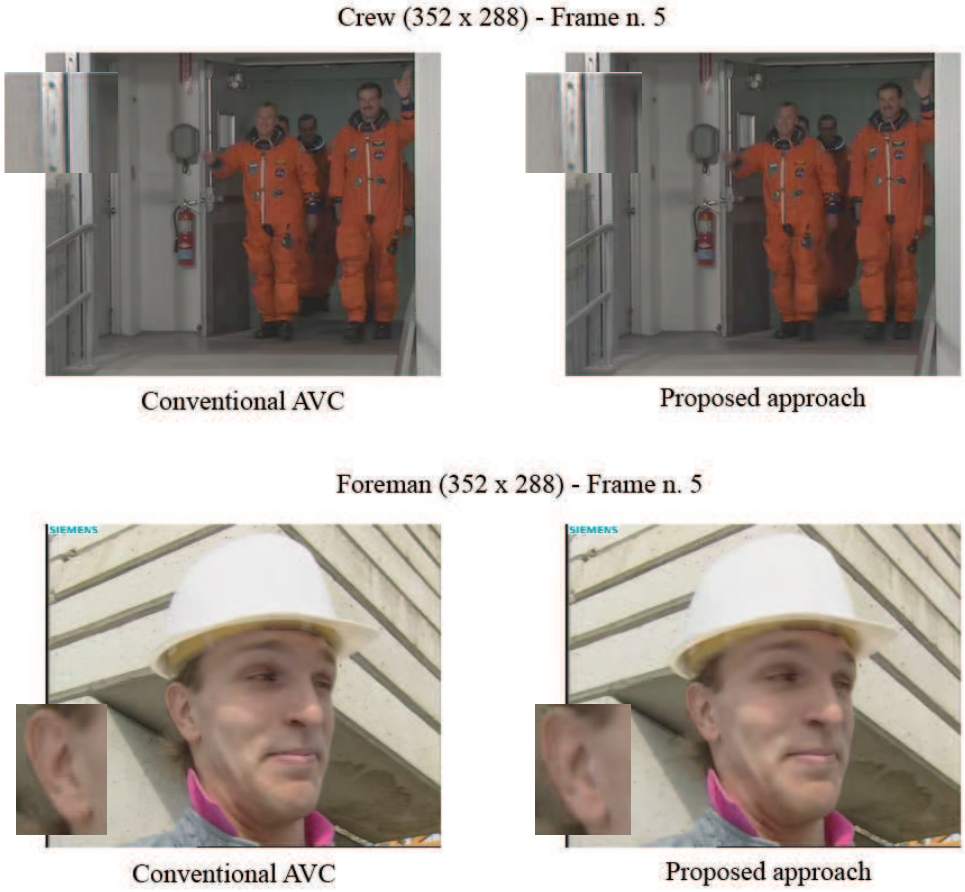



Figure 3.9: Visual comparison of proposed approach and conventional AVC for two frames of the Foreman and Crew sequences. Some details of the images are enlarged for better clarity.

Table 3-D: Proposed approach using different values of fixed threshold compared with the DF approach.

| **Sequence** | **Proposed** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | T=0.4 | | T=0.8 | | T=1.2 | | T=1.6 | | T=2.0 | | T=2.4 | | T=2.8 | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 1.87 | 60.3 | 3.79 | 56.6 | 3.64 | 60.8 | **3.38** | **60.8** | 4.92 | 61.1 | 6.41 | 66.6 | 6.97 | 79.2 |
| RaceHorses | 1.89 | 19.2 | 3.54 | 24.0 | **4.71** | **52.6** | 6.38 | 26.9 | 7.50 | 24.9 | 9.72 | 45.0 | 9.69 | 26.2 |
| Keiba | 0.97 | 9.7 | 3.08 | 16.1 | 2.51 | 28.5 | 3.66 | 17.2 | 4.15 | 16.2 | 5.60 | 14.9 | **3.58** | **26.4** |
| Bowing | 1.59 | 31.1 | 2.43 | 33.5 | 2.64 | 34.1 | **3.02** | **42.2** | 4.01 | 44.3 | 4.33 | 38.9 | 4.43 | 47.5 |
| Carphone | 3.10 | 30.9 | 4.35 | 38.9 | **5.37** | **41.7** | 6.54 | 37.4 | 7.30 | 48.8 | 8.59 | 44.1 | 9.79 | 48.6 |
| Crew | 4.93 | 32.7 | 8.27 | 38.0 | **8.99** | **49.8** | 12.17 | 46.3 | 14.24 | 43.7 | 15.37 | 51.9 | 16.47 | 56.1 |
| Foreman | 3.84 | 25.5 | 5.74 | 35.4 | **7.31** | **38.7** | 8.56 | 39.9 | 9.85 | 42.8 | 10.59 | 42.5 | 11.68 | 49.4 |
| Silent | 1.26 | 15.5 | **2.46** | **21.8** | 3.47 | 26.1 | 4.27 | 30.5 | 5.31 | 33.7 | 5.91 | 34.3 | 7.15 | 38.8 |
| Soccer | 2.16 | 16.3 | 4.22 | 23.8 | 5.99 | 30.7 | **7.62** | **32.9** | 9.33 | 38.6 | 10.80 | 41.1 | 12.56 | 41.5 |
| Akiyo | 1.04 | 22.9 | 1.16 | 32.6 | 1.02 | 35.1 | 1.97 | 41.9 | **2.65** | **40.0** | 2.98 | 45.0 | 3.46 | 42.4 |
| MotherAndD. | 1.34 | 15.2 | 3.31 | 24.4 | 5.10 | 31.0 | 6.82 | 36.7 | 9.30 | 38.3 | 11.37 | 42.7 | 13.67 | 48.9 |
| News | 0.79 | 0.7 | 1.42 | 9.6 | 1.82 | 9.7 | **2.00** | **15.4** | 3.31 | 17.1 | 3.12 | 19.5 | 3.07 | 26.6 |
| Suzie | 2.49 | 36.8 | **4.78** | **45.4** | 6.53 | 48.8 | 8.76 | 53.8 | 11.08 | 57.3 | 12.68 | 62.5 | 14.85 | 64.6 |
| Trevor | 1.86 | 12.0 | **3.76** | **21.7** | 4.64 | 26.8 | 5.59 | 31.9 | 6.54 | 32.9 | 7.74 | 36.2 | 8.19 | 39.2 |
| **Sequence** | **Proposed** | | | | | | | | | | | | **DF** | |
| | T=3.2 | | T=3.6 | | T=4.0 | | T=4.4 | | T=4.8 | | T=5.2 | | | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 7.97 | 70.3 | 7.82 | 56.5 | 8.56 | 80.8 | 8.86 | 72.4 | 10.32 | 66.5 | 10.08 | 60.9 | *3.49* | *54.9* |
| RaceHorses | 12.27 | 45.0 | 12.29 | 65.9 | 13.33 | 46.2 | 14.82 | 46.3 | 15.13 | 47.7 | 16.53 | 43.9 | *4.00* | *22.2* |
| Keiba | 5.16 | 24.8 | 5.58 | 26.8 | 6.53 | 26.6 | 7.68 | 29.8 | 7.40 | 31.4 | 8.18 | 28.6 | *2.79* | *19.6* |
| Bowing | 4.32 | 48.0 | 4.96 | 40.7 | 4.97 | 49.8 | 4.88 | 47.7 | 5.78 | 47.3 | 5.33 | 56.2 | *2.20* | *34.9* |
| Carphone | 10.35 | 49.4 | 11.32 | 54.3 | 11.99 | 53.0 | 13.00 | 47.7 | 13.46 | 59.7 | 14.66 | 57.9 | *5.21* | *39.3* |
| Crew | 18.78 | 53.5 | 19.37 | 53.5 | 22.13 | 55.9 | 21.24 | 51.6 | 21.81 | 58.7 | 23.12 | 55.8 | *9.01* | *35.4* |
| Foreman | 12.47 | 45.9 | 13.28 | 50.3 | 14.01 | 52.7 | 14.68 | 50.6 | 15.62 | 52.9 | 16.42 | 54.1 | *6.31* | *35.2* |
| Silent | 7.63 | 40.3 | 8.39 | 42.3 | 8.90 | 44.7 | 9.58 | 43.7 | 9.86 | 46.5 | 10.44 | 48.9 | *2.40* | *21.7* |
| Soccer | 13.86 | 44.4 | 15.15 | 46.8 | 16.38 | 48.8 | 17.54 | 51.2 | 18.41 | 51.9 | 19.56 | 51.9 | *6.92* | *30.1* |
| Akiyo | 4.31 | 48.1 | 4.82 | 52.0 | 5.79 | 51.4 | 5.87 | 52.8 | 6.78 | 51.7 | 7.59 | 55.5 | *1.78* | *34.5* |
| MotherAndD. | 15.73 | 53.4 | 18.13 | 52.4 | 19.36 | 52.5 | 21.80 | 55.2 | 22.75 | 62.4 | 24.41 | 63.1 | *3.99* | *26.9* |
| News | 3.56 | 28.6 | 4.36 | 28.4 | 4.82 | 36.8 | 5.07 | 33.0 | 5.56 | 36.5 | 6.00 | 34.1 | *1.13* | *14.3* |
| Suzie | 16.28 | 69.8 | 18.14 | 68.4 | 19.54 | 69.6 | 20.63 | 68.1 | 21.82 | 68.9 | 23.13 | 71.5 | *4.66* | *41.5* |
| Trevor | 9.65 | 45.3 | 11.09 | 50.5 | 12.00 | 45.5 | 12.76 | 49.6 | 13.95 | 51.0 | 15.22 | 52.4 | *3.47* | *18.9* |

Table 3-E: Proposed approach with thresholds estimated only at the beginning of each sequence.

| **Sequence** | **Proposed (Tested on 10 frames)** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $E_{target}=1$ | | $E_{target}=2$ | | $E_{target}=3$ | | $E_{target}=4$ | | $E_{target}=5$ | | $E_{target}=6$ | | $E_{target}=7$ | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 0.56 | 64.7 | 1.50 | 66.0 | 1.02 | 63.9 | 2.53 | 64.7 | 2.70 | 63.6 | 1.99 | 66.4 | 1.98 | 62.8 |
| RaceHorses | 0.99 | 12.5 | 1.23 | 14.7 | 1.58 | 10.1 | 2.55 | 17.1 | 2.89 | 19.5 | 3.16 | 17.4 | 3.77 | 24.0 |
| Foreman | 1.47 | 19.7 | 2.04 | 21.9 | 2.19 | 20.2 | 2.62 | 33.0 | 2.87 | 25.3 | 3.39 | 27.9 | 3.60 | 29.7 |
| Silent | 1.49 | 20.9 | 2.97 | 35.4 | 3.60 | 40.4 | 3.61 | 48.8 | 3.62 | 39.9 | 3.62 | 44.2 | 3.62 | 43.9 |
| MotherAndD. | 0.53 | 12.3 | 1.10 | 17.4 | 1.55 | 31.1 | 2.81 | 39.7 | 2.79 | 41.5 | 3.92 | 44.6 | 4.11 | 51.4 |
| Trevor | 0.39 | 5.7 | 0.70 | 5.2 | 1.18 | −8.3 | 2.20 | 31.0 | 3.28 | 21.2 | 4.52 | 27.5 | 5.23 | 28.9 |
| **Sequence** | **Proposed (Tested on 20 frames)** | | | | | | | | | | | | | |
| | $E_{target}=1$ | | $E_{target}=2$ | | $E_{target}=3$ | | $E_{target}=4$ | | $E_{target}=5$ | | $E_{target}=6$ | | $E_{target}=7$ | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 0.40 | 69.0 | 0.33 | 68.6 | 0.48 | 70.9 | 2.61 | 69.5 | 2.83 | 70.3 | 2.00 | 70.1 | 2.02 | 69.7 |
| RaceHorses | 1.42 | 10.6 | 1.77 | 13.5 | 1.92 | 15.6 | 3.97 | 19.1 | 4.02 | 20.5 | 4.64 | 22.1 | 5.67 | 25.4 |
| Foreman | 2.77 | 19.9 | 3.18 | 21.1 | 3.71 | 24.8 | 4.19 | 27.4 | 4.55 | 28.6 | 5.28 | 30.7 | 5.45 | 32.8 |
| Silent | 1.78 | 24.8 | 4.47 | 35.4 | 5.53 | 39.9 | 5.36 | 46.1 | 5.35 | 45.2 | 5.35 | 47.1 | 5.35 | 46.1 |
| MotherAndD. | −0.49 | 5.9 | 0.06 | 22.1 | 1.18 | 42.2 | 3.08 | 49.7 | 3.69 | 43.4 | 5.03 | 56.8 | 5.28 | 58.9 |
| Trevor | 0.70 | 6.3 | 1.34 | 26.8 | 2.08 | 29.3 | 3.44 | 37.1 | 5.12 | 36.2 | 6.13 | 31.1 | 7.46 | 42.0 |
| **Sequence** | **Proposed (Tested on 50 frames)** | | | | | | | | | | | | | |
| | $E_{target}=1$ | | $E_{target}=2$ | | $E_{target}=3$ | | $E_{target}=4$ | | $E_{target}=5$ | | $E_{target}=6$ | | $E_{target}=7$ | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 1.27 | 63.5 | 2.89 | 63.8 | 2.06 | 63.4 | 2.82 | 65.2 | 4.95 | 65.8 | 3.17 | 65.5 | 2.26 | 66.0 |
| RaceHorses | 0.83 | 11.7 | 1.74 | 15.1 | 2.30 | 18.7 | 4.05 | 22.7 | 4.57 | 24.2 | 6.17 | 29.8 | 8.01 | 30.6 |
| Foreman | 3.04 | 21.5 | 3.29 | 23.3 | 4.05 | 25.9 | 4.46 | 28.4 | 4.98 | 31.7 | 5.60 | 32.0 | 6.28 | 29.6 |
| Silent | 2.74 | 22.2 | 6.39 | 36.4 | 8.12 | 43.2 | 8.24 | 46.5 | 8.17 | 47.0 | 8.17 | 47.1 | 8.17 | 45.9 |
| MotherAndD. | 1.74 | 19.8 | 3.97 | 27.5 | 8.11 | 42.9 | 14.36 | 50.3 | 17.59 | 59.7 | 23.29 | 55.9 | 23.73 | 62.1 |
| Trevor | 0.97 | 13.1 | 1.82 | 16.4 | 2.89 | 18.1 | 3.88 | 23.8 | 6.28 | 24.9 | 8.33 | 27.2 | 10.37 | 36.8 |
| **Sequence** | **Proposed (Tested on full length of the sequences)** | | | | | | | | | | | | | |
| | $E_{target}=1$ | | $E_{target}=2$ | | $E_{target}=3$ | | $E_{target}=4$ | | $E_{target}=5$ | | $E_{target}=6$ | | $E_{target}=7$ | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 2.46 | 53.6 | 4.04 | 59.1 | 4.73 | 60.9 | 7.43 | 63.7 | 8.01 | 64.2 | 7.33 | 63.6 | 7.18 | 64.7 |
| RaceHorses | 1.34 | 15.7 | 2.27 | 18.7 | 3.49 | 22.8 | 4.13 | 24.7 | 4.96 | 28.1 | 6.20 | 31.0 | 7.69 | 33.8 |
| Foreman | 2.86 | 20.4 | 2.89 | 22.5 | 3.52 | 24.9 | 4.26 | 26.9 | 4.38 | 30.7 | 5.11 | 31.4 | 5.67 | 33.2 |
| Silent | 3.39 | 21.9 | 7.98 | 37.9 | 10.07 | 43.8 | 10.19 | 47.6 | 10.26 | 47.9 | 10.26 | 47.6 | 10.26 | 48.4 |
| MotherAndD. | 1.86 | 15.2 | 3.86 | 25.2 | 8.29 | 37.6 | 14.32 | 51.6 | 18.98 | 56.2 | 24.00 | 60.5 | 24.53 | 58.7 |
| Trevor | 3.16 | 17.2 | 5.04 | 24.5 | 5.48 | 27.2 | 7.18 | 35.3 | 10.23 | 43.1 | 11.85 | 43.1 | 14.77 | 48.2 |

## 3.5 Conclusions

Fractional precision motion estimation is a valuable tool for improving compression efficiency of modern video coders. On the other hand the interpolation and additional search required by the method add a consistent amount of computational complexity to the encoding.

A novel method for fast sub-pixel motion estimation in the context of the AVC standard was presented in this chapter. The approach aims to reduce computational complexity by skipping the derivation of motion vector fractional refinements in blocks where performing sub-pixel motion estimation would not considerably improve the compression efficiency. An original geometrical characterisation of the residual error surface was proposed, referred to as sampled curvedness, based on the information available at the encoder side after integer-precision motion estimation.

The method considers then a binary classifier that takes as input the sampled curvedness and consequently selects whether sub-pixel motion estimation is needed in the current block or not. Several approaches to derive the thresholds used by the classifier are also proposed in the chapter. Instead of being based simply on the observation of blocks where sub-pixel motion estimation is used, the computation of the thresholds is based on the impact of fractional refinements on each block in terms of the actual reduction of prediction error due to using fractional motion vector components with respect to the integer valued solution.

Along with a solution where a fixed threshold is used throughout the encoding, a method is also proposed in which the thresholds are updated on line during the encoding. The relationship between the threshold values and the corresponding fractional impact is estimated at the encoder based on an initial group of training blocks. This is used to compute an initial estimate of the threshold. Also, the thresholds are successively updated in case the local coding conditions in the sequence change drastically with

respect to the blocks used for the training.

The approach is shown obtaining consistent time savings with respect to conventional AVC, while at the same time limiting the losses in compression efficiency. Consistent time savings are obtained in all tests. The approach is capable of mitigating the effects in terms of performance losses: higher time savings are obtained when fractional refinements provide small impacts on the coding performance, whereas acceptable losses are obtained in sequences where sub-pixel motion estimation has a high impact on the compression efficiency.

Finally it should be noted that while in this thesis the approach is only implemented and tested in the context of the AVC standard, in theory it could also be applied to HEVC. Note though that this would require considerable adaptation. In particular, most HEVC encoder implementations (such as that used in the HM reference software [85]) make use of a fast integer-precision motion estimation algorithm referred to as TZSearch. When using such algorithm not all locations in the window at integer-precision are tested, following similar schemes as the EPZS algorithms illustrated in Chapter 2. As a result, the distortion values at the locations necessary to compute the SC using Equations 3.3-3.6 may not be available. Different techniques could be proposed to solve this issue: for instance, the missing distortion values may be replaced with other values estimated based on the available information; or the expression to compute the sampled curvedness could be adapted to the case when not all distortion values are available. These modifications would drastically change the framework of the approach proposed in this chapter and fall out of the scope of this thesis.

Table 3-F: Proposed approach with on-the-fly threshold updating.

| **Sequence** | **Proposed (adaptive threshold)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $E_{target} = 1$ | | $E_{target} = 2$ | | $E_{target} = 3$ | | $E_{target} = 4$ | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 1.97 | 41.6 | 2.00 | 42.7 | 2.31 | 44.0 | 3.63 | 47.4 |
| RaceHorses | 1.36 | 9.1 | 1.53 | 12.5 | 2.14 | 11.2 | 2.43 | 16.0 |
| Keiba | 2.47 | 5.5 | 1.35 | 12.7 | 2.52 | 10.1 | 2.39 | 14.3 |
| Bowing | 0.44 | 10.2 | 1.24 | 21.3 | 1.83 | 28.0 | 2.39 | 23.3 |
| Carphone | 1.76 | 22.7 | 2.20 | 21.0 | 2.67 | 24.8 | 2.90 | 28.9 |
| Crew | 3.92 | 14.0 | 3.00 | 18.9 | 2.98 | 17.3 | 3.74 | 18.5 |
| Foreman | 1.90 | 12.6 | 2.71 | 18.4 | 3.22 | 15.6 | 3.77 | 23.5 |
| Silent | 1.01 | 8.6 | 1.62 | 5.4 | 2.66 | 16.1 | 3.70 | 24.2 |
| Soccer | 1.47 | 0.4 | 2.62 | 6.2 | 3.43 | 13.9 | 3.79 | 21.3 |
| Akiyo | 1.70 | 41.1 | 2.68 | 45.7 | 2.52 | 43.5 | 2.65 | 40.9 |
| MotherAndD. | 0.30 | 5.9 | 1.91 | 17.5 | 3.00 | 21.5 | 4.01 | 28.3 |
| News | 0.90 | 8.6 | 1.54 | 12.7 | 2.27 | 17.5 | 2.59 | 21.8 |
| Suzie | 0.71 | 23.6 | 1.38 | 26.1 | 2.23 | 32.9 | 2.52 | 31.2 |
| Trevor | 1.84 | 5.2 | 2.17 | 14.5 | 2.32 | 17.1 | 2.84 | 21.3 |
| *Mean* | 1.55 | | 2.00 | | 2.58 | | 3.10 | |
| *Variance* | 0.86 | | 0.34 | | 0.20 | | 0.40 | |
| **Sequence** | **Proposed (adaptive threshold)** | | | | | | **Fixed threshold** | |
| | $E_{target} = 5$ | | $E_{target} = 6$ | | $E_{target} = 7$ | | T=1.6 | |
| | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime | BD-rate | ΔTime |
| Mobisode2 | 3.98 | 54.2 | 4.81 | 51.8 | 5.25 | 47.4 | 3.38 | 60.8 |
| RaceHorses | 3.70 | 19.5 | 4.09 | 24.0 | 4.54 | 20.9 | 6.38 | 26.9 |
| Keiba | 3.25 | 16.1 | 3.75 | 8.0 | 3.80 | 13.9 | 3.66 | 17.2 |
| Bowing | 2.76 | 28.5 | 3.04 | 37.0 | 3.52 | 35.2 | 3.02 | 42.2 |
| Carphone | 3.67 | 25.5 | 4.63 | 28.3 | 5.54 | 34.6 | 6.54 | 37.4 |
| Crew | 4.98 | 23.7 | 5.39 | 20.0 | 6.81 | 23.8 | 12.17 | 46.3 |
| Foreman | 4.19 | 19.5 | 5.08 | 25.4 | 5.75 | 31.4 | 8.56 | 39.9 |
| Silent | 4.64 | 29.8 | 5.38 | 34.9 | 5.93 | 32.3 | 4.27 | 30.5 |
| Soccer | 4.14 | 19.6 | 4.85 | 18.7 | 6.02 | 27.7 | 7.62 | 32.9 |
| Akiyo | 2.90 | 37.9 | 3.12 | 43.0 | 3.98 | 37.7 | 1.97 | 41.9 |
| MotherAndD. | 4.59 | 34.5 | 5.78 | 32.4 | 6.30 | 36.5 | 6.82 | 36.7 |
| News | 2.84 | 21.7 | 3.01 | 22.4 | 3.40 | 24.2 | 2.00 | 15.4 |
| Suzie | 3.82 | 37.2 | 4.60 | 39.2 | 5.44 | 40.7 | 8.76 | 53.8 |
| Trevor | 3.34 | 23.6 | 3.79 | 25.7 | 4.93 | 24.7 | 5.59 | 31.9 |
| *Mean* | 3.77 | | 4.38 | | 5.02 | | 5.32 | |
| *Variance* | 0.49 | | 0.85 | | 1.42 | | 8.60 | |

# Chapter 4

# Enhanced Inter-Prediction

The proposed enhanced inter-prediction (EIP) is presented here, as a module that can be included in typical video coding schemes to improve compression efficiency by means of on-the-fly parametric transformations of the prediction candidates.

## 4.1 Transformation of the reference frames

### 4.1.1 Motivation

Video coding schemes are typically formed of a succession of steps. Among all these steps the inter-prediction module is in many cases the most beneficial part of a conventional video coding scheme, namely the part that returns the highest compression ratio between the input content and the output data [107]. Prediction relies on the assumption that high amounts of redundancy can be found in the original signal prior to compression. Information is extracted from previously encoded parts of the sequence, and used to decrease the energy of the data to encode next. Thanks to the high presence of temporal redundancy, the data output by the inter-prediction module may be formed of a number of residuals with very low energy, plus some side parameters such as the

motion information.

The way the information is extracted from previous frames and used to compute a prediction may depend on encoder decisions and characteristics inherent to the codec or compression scheme being used. For instance in the case of integer-precision unidirectional motion estimation, the information in the reference frames is just copied as prediction for the current block. While it is a simple and yet successful technique, this is clearly a sub-optimal approach which has been extended and enhanced in many ways.

For instance manipulations of reference frames have been successfully used in image and video coding to improve inter-prediction, as already illustrated in previous parts of this thesis. A portion of the signal is transformed according to some predefined methods before being considered for prediction. This is the case for instance when using sub-pixel motion estimation, global motion estimation, or weighted prediction. In all these cases, previously encoded frames are transformed before being included in the list of reference frames, with the goal of reducing as much as possible the prediction error. The encoder choice is limited to selecting whether to use this transformed reference frame or use the original one.

Formally such methods can be illustrated as follows. Assume that a certain set of tools is taken into consideration by the encoder, each denoted as $T_k$. This might include sub-pixel motion estimation, global motion estimation, weighted prediction or any similar method based on reference transformation. Consider that a certain frame $I$ is input to the encoder, and that inter-prediction is allowed on the blocks within $I$ with an associated list (or lists) of reference frames. All reference frames are denoted here as $R_i$ regardless of which list they are extracted from. The encoder considers a tool $T_k$ and applies the corresponding transformation to a reference frame $R_i$ to obtain a transformed reference $\bar{R}_{i,k}$. Such transformation may be based on the values of $R_i$ alone (as is the case for sub-pixel motion estimation interpolation), or may also include the content of the current frame $I$ (as is typically the case for global motion estimation or weighted prediction). The transformed references $\bar{R}_{i,k}$ are then considered during the

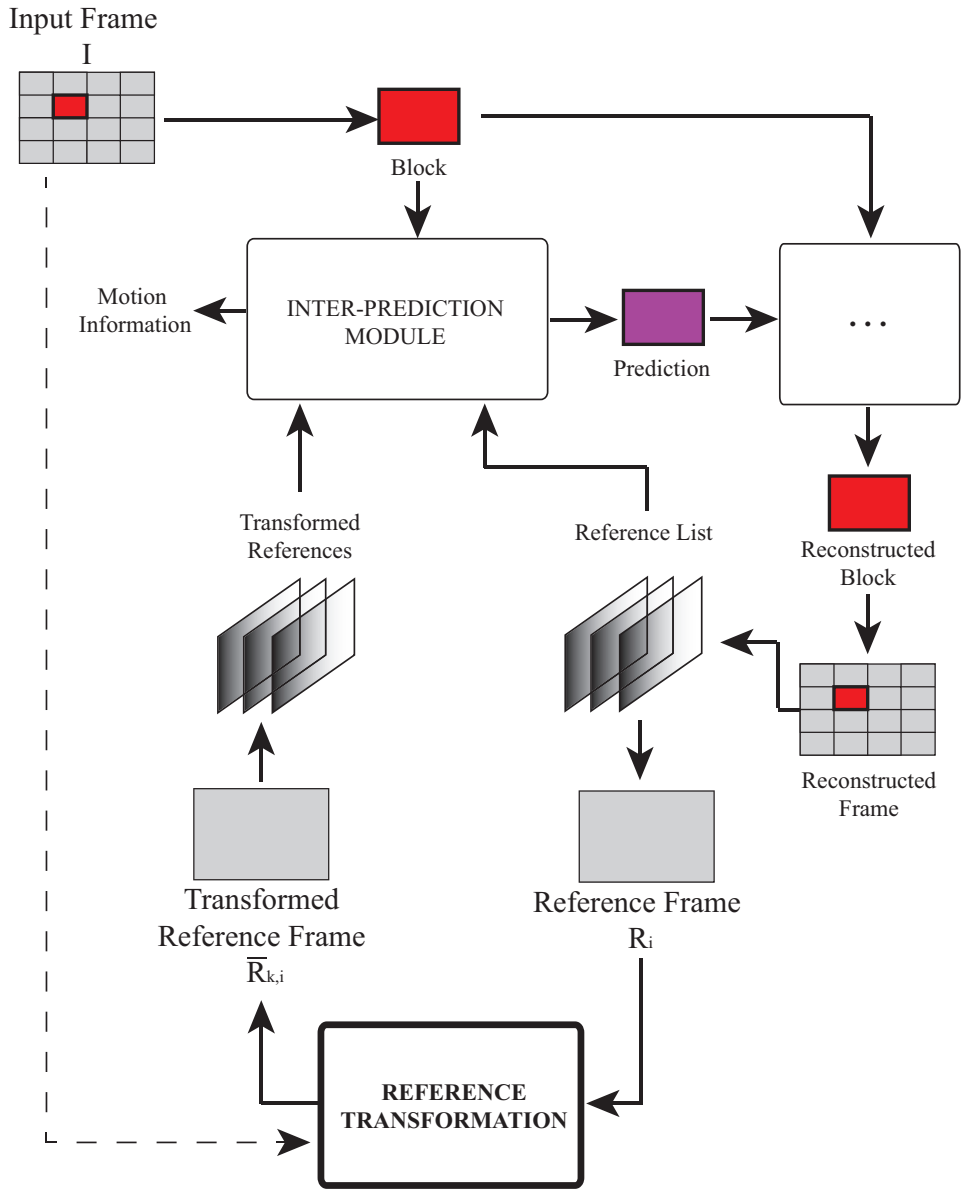



Figure 4.1: Block diagram of an encoder using conventional reference transformation methods. Only blocks of interest to reference transformation are shown in the diagram, all other blocks are omitted.

encoding of each block as if they were additional reference frames included in the list. When performing RD optimisation, the encoder selects whether to use such a tool on a block in the same way as when choosing which reference frame to use. This scheme is illustrated in Figure 4.1.

This approach has some strong limitations. The transformations that can be applied

to a reference $R_i$ to obtain $\bar{R}_{i,k}$ take only into consideration global characteristics of the frame and do not depend on the local content of the current block being encoded. For this reason methods such as global motion estimation or weighted prediction are only effective in particular cases (e.g. fades or brightness variations), and typically do not provide benefits under more common coding conditions, as shown in the rest of this chapter.

Other methods have been proposed which instead consider transformations at a block level. The affine motion model is a well-known example of such methods, investigated in the context of video coding applications since the mid 90s [108]. Such model is an extension to the translational model considered in conventional motion compensation. Instead of considering only 2 parameters to transmit the displacement information (the motion vector components), 6 parameters are transmitted to represent also information regarding scaling or rotation of the prediction block. The problem with the affine model is that it requires transmission of a high number of parameters for each block, drastically limiting the RD performance of the prediction unit. In fact motion compensation based on the affine model has been rarely used in video coding standards. The method was recently applied and studied in the context of HEVC [109]. Due to the average larger size of the blocks in this standard with respect to its predecessors the affine model achieves better results. Still the approach was shown improving the coding efficiency only in particular sequences which present high amounts of non-translational motion. Finally, string matching was also proposed [110], as a method in which a block is further split into regions of arbitrary shape, which are then rearranged into one-dimensional strings. A match for each string is then extracted from the buffer of previously encoded and reconstructed samples. A parameter to represent the displacement between the two strings is transmitted for each string. Such method is only suited for screen content coding and does not bring benefits under other conditions.

### 4.1.2 Enhanced Inter-prediction

All techniques presented in the previous sub-section are based on the assumption that inter-prediction can be improved by improving motion compensation. Such improvements may be indirectly produced as an effect of changing the content in the reference frames, or directly produced by changing the way motion compensation is performed.

The methods presented in this chapter are based on a different idea. Instead of targeting improvements of the motion compensation module, inter-prediction can be enhanced by complementing motion compensation with a completely independent additional tool. Such tool would act at the same time as motion estimation, enhancing each prediction candidate with the goal of returning a new solution which is better suited for the local content currently being encoded.

In particular while performing motion estimation the encoder considers a set of motion vectors $MV_0, MV_1, ...MV_L$ where $MV_i = (m_i, n_i)$. The content of the set and the number $L$ of its elements depend on the motion estimation algorithm being used. Each motion vector points to a location in a previously encoded reference frame. The encoder extracts from a reference frame at this location a prediction block $P_i$ of the same size as the current block $X$. Denote as $M$ and $N$ the height and width of this block respectively. While at this stage typical encoders consider a cost $\Delta(P_i, X)$ computed in an RD sense, assume for simplicity that only the distortion calculated using SAD is considered when computing $\Delta(P_i, X)$. Finally the encoder selects the motion vector $MV^o$ that produces the minimum distortion, or $MV^o = MV_k$ such that:

$$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|P_k(m,n) - X(m,n)| \leq \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|P_i(m,n) - X(m,n)|, \forall i : i = 0, ..., L-1.$$

Denote the corresponding optimal prediction block as $P_k = P^o$.

Consider now a certain parametric matrix function $\Theta_w(\bullet)$, where $w = [w_0, w_1, ..., w_{T-1}]$ is a vector of parameters of length $T$. While the expression "matrix function" may have

different meanings, in the rest of this chapter it is used according to the following definition [111]: for a given set of parameters, $\Theta_w$ is a scalar function defined in the domain $\mathbb{N}^{M\times N}$, which maps a matrix $Y$ of size $M \times N$ to another matrix $Z$ also of size $M \times N$.

Assume now that there exist a certain set of parameters $w_t$ and a prediction candidate $P_j$ such that the matrix $EIP_j = \Theta_{w_t}(P_j)$, obtained by applying the function to the candidate, satisfies:

$$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} |EIP_j(m,n) - X(m,n)| \leq \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} |P^o(m,n) - X(m,n)|,$$

that is, the distortion between the matrix $EIP_j$ and the original block $X$ is lower than that obtained with the previously considered optimal solution $P^o$. Given that it is provided with the array of parameters $w_t$, the inter-prediction module can make use of $EIP_j$ instead of $P^o$ as the optimal prediction for $X$. Due to the smaller residual samples a higher compression efficiency can be expected.

The parametric transformation $\Theta_w(\bullet)$ can be considered as a part of an additional module which is included in both the encoder and decoder side, which is referred to as the enhanced inter-prediction (EIP) in the rest of this thesis. A block diagram of the proposed modified encoder with EIP is shown in Figure 4.2. The transformation of a prediction candidate is performed in the modified encoder directly inside the loop among prediction candidates during motion estimation. As a consequence the associated transformations are able to influence the outcomes of motion estimation on-the-fly. For each prediction candidate $P_i$ obtained with $MV_i$, the optimal set of parameters $w$ is computed such that the distortion between original block and enhanced prediction is minimised. Finally, the optimal solution comprising of motion vector $MV^o$ and parameters $w^o$ is output. These are entropy coded in the bitstream and sent to the decoder side. The decoder extracts the motion information and EIP parameters, and performs motion compensation followed by enhanced inter-prediction, to eventually obtain the appropriately enhanced prediction block.

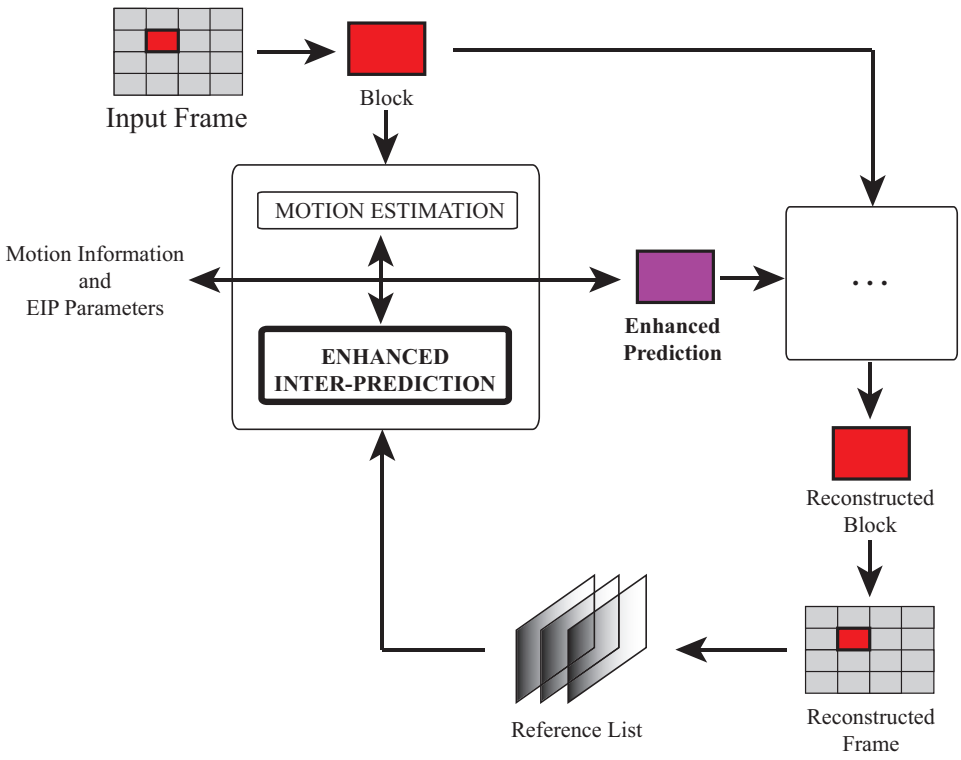



Figure 4.2: Block diagram of an encoder using the proposed Enhanced Inter-prediction. Only blocks of interest to the approach are shown in the diagram, all other blocks are omitted.

However in order to achieve the compression efficiency needed by most applications, video coding schemes must refer to RD theory when computing the cost $\Delta(P_i, X)$. The distortion between $P_i$ and $X$ is adjusted with an estimate of the rate to transmit $MV_i$ and/or other data in the motion information. Clearly using the same cost to evaluate a block $P_i$ or an enhanced block $EIP_j$ is not optimal, in that such a metric does not take into consideration the rate needed to transmit the parameters $w_t$. A different solution must be considered to take into account this rate which depends on the number and characteristics of the parameters in $w$, as illustrated in the following of this chapter.

## 4.2 The Shifting Transformation

### 4.2.1 Definition and Derivation of the Shift Parameter

The insertion of the EIP module in the encoder and decoder loops adds great flexibility to the type of prediction that can be generated. Such flexibility may be used to target specific applications or instead to improve compression efficiency in the broadest range of conditions. This second objective was targeted in this section.

Moreover, while in theory many transformations could be investigated, in practice the EIP can only be applied in a meaningful way in a video coding scenario if the involved parametric matrix function satisfies the following requirements: (i) the derivation of the parameters needed for a certain prediction candidate should have the smallest possible impact on the encoder computational complexity (ii) the complexity of applying the function itself to a given matrix should be small both at the encoder and decoder side (iii) the length of the vector $w$ which needs to be transmitted in the bitstream should be as small as possible, and (iv) $w$ should be easy to compress.

Finding transformations capable of meeting such requirements with the goal of improving compression efficiency is not trivial. Under the light of these observations, in this chapter the EIP transformations are defined with the specific goal of compensating for differences between inter-predicted and original signals which affect the entire block. For instance the function $\Theta_w$ could be defined as a linear transformation:

$$\Theta_{a,s}\left(P\right) : \boldsymbol{eip}(m,n) = \alpha p(m,n) + s, \tag{4.1}$$

which requires only two parameters, or $w = [\alpha, s]$ where $\alpha$ is a non-negative integer and $s$ is an integer. By appropriately defining the parameters $\alpha$ and $s$, the inter-predicted blocks can be appropriately scaled and shifted. Such transformation meets most of the aforementioned requirements: it is relatively fast to apply, it only requires transmission of two parameters, and such parameters are relatively easy to compress similar to the

transmission of the motion vector difference components.

Unfortunately the derivation of $\alpha$ and $s$ for each motion vector in the list during motion estimation is instead very complex. For this reason the particular case when $\alpha$ is fixed to 1 is considered. Such parametric matrix function is referred to here as the shifting transformation, defined by a single integer parameter as:

$$\Theta_s(P) : \boldsymbol{eip}(m,n) = p(m,n) + s. \tag{4.2}$$

A graphical representation of a typical situation where the shifting transformation might reduce the prediction error obtained with a particular prediction candidate is shown in Figure 4.3. A certain prediction candidate is considered for the original block (shown at the top of the figure). On the left, the prediction samples without any shift are shown along with the corresponding residual samples. On the right, the prediction samples transformed by means of a certain shift (represented in black) are represented. The corresponding residual samples are smaller in absolute value, consequently resulting

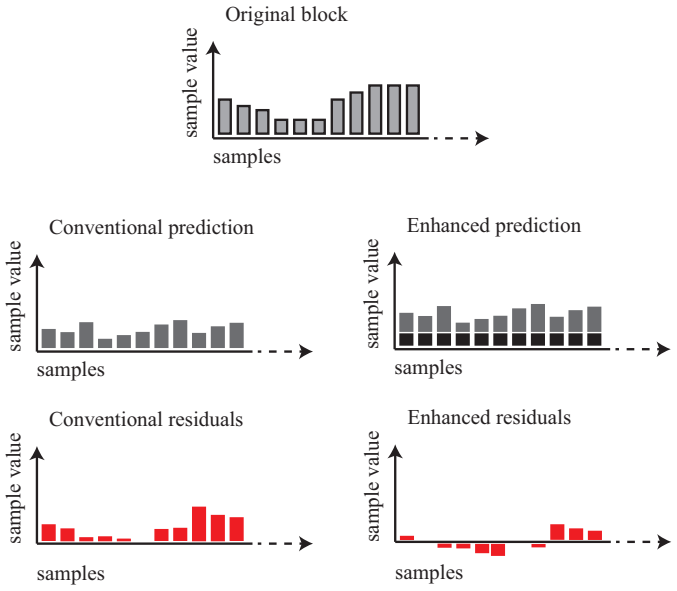



Figure 4.3: Example of a successful application of the shifting transformation.

in a lower prediction error.

In the particular case of the shifting transformation, the derivation of the optimal parameter $s$ can be performed by means of a fast algorithm, as illustrated here. The method is defined using SAD as the distortion measure. At this stage only the prediction distortion is considered in the cost $\Delta(P, X)$ (other measures could be used, but the algorithm would need some adaptation). The method outputs, in a single step, the optimal shift and the corresponding gain in terms of SAD provided by the transformation of the prediction using this shift.

Consider the residual block $R$ obtained as the difference between a prediction block $P$ and the current block $X$. Assume all elements in each of these three blocks are arranged in column vectors of length $B = M \times N$, where $M$ and $N$ are the block height and width respectively. Refer to the elements in each of these three vectors as $p(i)$, $x(i)$ and $r(i) = p(i) - x(i)$, where $i = 0, 1, ..., (B-1)$.

The SAD between the original prediction block $P$ and $X$ is:

$$SAD\left(P, X\right) = \sum_{i=0}^{B-1} \left|r\left(i\right)\right| = \sum_{i=0}^{B-1} \left|p\left(i\right) - x\left(i\right)\right|. \tag{4.3}$$

Consider now with no loss of generality that the elements in the vector $R$ are rearranged in increasing order, i.e. $r(i) \leq r(j), \forall i \leq j$. Clearly this rearrangement does not affect the SAD. Denote now with $N_{+}\left(0\right)$, $N_{0}\left(0\right)$ and $N_{-}\left(0\right)$ the number of positive, zero and negative elements in $R$. The numbers $N_{+}$,$N_{-}$ and $N_{0}$ are referred collectively as the sign ratio of $R$ in the rest of this chapter.

Applying the shifting transformation on the prediction $P$ with a unitary positive shift $s = +1$ results in a new vector of residual samples $R_{+1}$ with elements $r_{+1}\left(i\right) =$

$(p(i)+1)-x(i)$. The corresponding SAD, denoted as $SAD(+1)$, is obtained as:

$$\begin{aligned} SAD(+1) &= \sum_{i=0}^{B-1} |r(i)+1| \\ &= -\sum_{i=0}^{N_-(0)-1} (r(i)+1) + \sum_{i=N_0(0)}^{B-1} (r(i)+1) \\ &= SAD(0) - N_-(0) + N_0(0) + N_+(0). \end{aligned}$$

If the factor $(-N_-(0)+N_0(0)+N_+(0))$ at the right hand side of this equation is negative, the corresponding prediction error obtained by transforming the block $P$ with a parameter $s=+1$ decreases (namely $SAD(+1) < SAD(0)$). This happens if and only if the number of negative elements in $R$ is larger than the remaining number of zero and positive elements, that is if the following condition is satisfied:

$$N_-(0) > N_0(0) + N_+(0). \tag{4.4}$$

The additional factor $(-N_-(0)+N_0(0)+N_+(0))$ is in fact equal to the offset between the non-enhanced SAD and the enhanced SAD.

Consider now that Condition 4.4 is satisfied for the current $P$ and $X$. Denote with $k = r(N_-(0)-1)$ the maximum negative element in $R$. Following from the assumption that elements in $R$ are sorted in increasing order, this is also the value of the last negative element in the vector. Then, by applying any positive shift $s>0$ and such that $s<|k|$, Condition 4.4 would still be satisfied:

$$N_-(+s) > N_0(+s) + N_+(+s).$$

The offset between the non-enhanced SAD and the enhanced SAD would be equal

to:

$$\Delta\left(SAD\right) = s \cdot \left(-N_{-}\left(0\right) + N_{0}\left(0\right) + N_{+}\left(0\right)\right).$$

By applying a shift exactly equal to $s = |k| = |r\left(N_{-}\left(0\right) - 1\right)|$, an additional zero element would correspondingly appear in the shifted residual vector $R_{+s}$, or:

$$\begin{aligned} N_{0}\left(+s\right) &= 1, \\ N_{+}\left(+s\right) &= N_{0}\left(0\right) + N_{+}\left(0\right), \\ N_{-}\left(+s\right) &= N_{-}\left(0\right) - 1. \end{aligned}$$

The problem becomes then that of verifying whether an even larger shift than $s = |k|$ can be applied, still resulting in a decrease in distortion. In order for this to happen the new sign ratio in the shifter residual vector $R_{+s}$ must satisfy condition 4.4. This can be written in terms of the original sign ratio for the non-shifted residual vector $R$ as:

$$N_{-}\left(0\right) > N_{0}\left(0\right) + N_{+}\left(0\right) + 2. \tag{4.5}$$

If the inequality in Equation 4.5 is not satisfied, a larger shift than $s = |k| = |r\left(N_{-}\left(0\right) - 1\right)|$ does not reduce the SAD, which means $s$ is the shift that provides the maximum possible distortion reduction.Such reduction can be easily computed as:

$$\left(r\left(N_{-}\left(0\right) - 1\right) \cdot \left(-N_{-}\left(0\right) + N_{0}\left(0\right) + N_{+}\left(0\right)\right)\right),$$

referred here to as the SAD offset obtained with shift $s$.

Conversely if the inequality in Equation 4.5 is satisfied, the distortion can be further decreased with a shift larger than $|k|$. The shift can be increased until Condition 4.4 is satisfied.

This process can be iterated to formulate a one-step algorithm capable of providing the optimal value of the shift which can be applied to provide the maximum possible SAD offset. In particular given a certain original block $X$ and a prediction candidate $P$, the sorted residual vector $R$ is computed and the corresponding sing ratio is derived. Then, if Condition 4.4 is satisfied, the optimal shift to obtain the maximum SAD offset can be obtained as:

$$s = -r\left(N_{-}\left(0\right) - n_{max}\right), \tag{4.6}$$

where $n_{max}$ is the maximum integer number $n$ such that:

$$N_{-}\left(0\right) > N_{0}\left(0\right) + N_{+}\left(0\right) + 2n, \tag{4.7}$$

or:

$$n_{max} = \left\lfloor \frac{N_{-}\left(0\right) - N_{0}\left(0\right) - N_{+}\left(0\right)}{2} \right\rfloor. \tag{4.8}$$

The SAD offset obtained as a result of using the shift in Equation 4.6 is:

$$\begin{aligned} \Delta\left(SAD\right) &= SAD\left(0\right) - SAD\left(-r\left(N_{-}\left(0\right) - n_{max}\right)\right) = \\ &= \sum_{n=0}^{n_{max}-1} -\left(N_{0}\left(0\right) + N_{+}\left(0\right) - N_{-}\left(0\right) + 2n\right) \times \\ &\times \left[r\left(N_{-}\left(0\right) - n - 1\right) - r\left(N_{-}\left(0\right) - n\right)\right]. \end{aligned} \tag{4.9}$$

Equations 4.6, 4.8 and 4.9 can be used to obtain the optimal positive shift parameter and corresponding SAD offset in a single step from a pair of original block and prediction candidate in case Condition 4.4 is satisfied. Equivalent expressions can be defined following a similar derivation for the case when a negative shift parameter is required.

In particular it is easy to prove that a negative shift parameter can be successfully applied to decrease the SAD if and only if the following conditions is satisfied:

$$N_{=}(0) > N_0(0) + N_{-}(0). \tag{4.10}$$

Note that Condition 4.10 and Condition 4.4 are mutually exclusive. Under the assumption that Condition 4.10 is satisfied, then the optimal negative shift can be obtained as:

$$s = r\left(N_0(0) + N_{+}(0) + n_{max}\right), \tag{4.11}$$

where $n_{max}$ is defined as in 4.8.

### 4.2.2 Shifts in Rate-Distortion Optimisation

Assume that a set of motion vectors $MV_0, MV_1, ...MV_L$ is considered by the encoder while coding block $X$, where $MV_i = (m_i, n_i)$. A shift $s_i$ is computed for each motion vector along with the corresponding SAD offset $\Delta_i(SAD)$. Clearly, $\Delta_i(SAD) = 0$ if $s_i = 0$.

A modified RD cost function $J$ can be defined to also consider the impact of transmitting the shift, or:

$$J_i = SAD(s_i) + \lambda_{MV} R_{MV_i} + \lambda_s R_{s=s_i}. \tag{4.12}$$

Equation 4.12 is an extension of the conventional RD optimization Lagrangian cost. It makes use of two Lagrangian multipliers for motion vectors and shift values respectively, to account for their respective costs of transmission $R_{MV_i}$ and $R_{s=s_i}$.

### 4.2.3 Implementation in AVC

The EIP with shift transformation was implemented in the AVC reference software (version 18.2 [72]). I slices, intra-predicted macroblocks and the SKIP mode are left unchanged. In case a macroblock is inter-predicted using multiple partitions, the optimal shift is computed for each partition along with the rest of the motion information.

The reference software was modified to accommodate the EIP in a modular way so that when the module is not used no effects are produced and the coding is identical to the original JM. Some experiments were first performed in order to adapt the encoder to the EIP and also define the multiplier $\lambda_s$. As an example the shift values found in all the optimal inter-prediction solutions using the algorithm in subsection 4.2.1 while coding a small portion of a test sequence are represented in Figure 4.4. The figure shows the histogram of the frequency of occurrence of each value. The distribution in the figure is clearly centred in zero with a low deviation: even without considering RD cost, still 14% of the prediction blocks are found to have an optimal shift equal to zero. Also, large shifts are in general very rare with the vast majority (93%) being in the range $-20 \leq s \leq 20$. Following from these observations, the EIP module was implemented to take advantage of this strong characterisation of the shift distribution. In particular RD optimization was adapted accordingly, and variable length coding was used for entropy coding.

The distribution shown in Figure 4.4 very closely resembles the distribution of the optimal motion vector differences resulting from typical video coding schemes. Such distribution is also strongly centred in zero, and motion vector differences also tend to assume relatively small values but may exceptionally assume very large ones. Due to such similarities the same Lagrangian multiplier used to account for the motion vector difference was used to account for the shift parameter, or $\lambda_s = \lambda_{MV}$ in Equation 4.12. Also due to such a strong characterisation, the encoder was modified to keep track of a non-enhanced solution regardless of the shift parameters output by the algorithm.

In particular the solution when $s = 0$ in all candidates $MV_0, MV_1, ...MV_{L-1}$ is always considered in an RD sense along with the enhanced solution. Note that a rate cost is associated with a zero-valued shift $s = 0$ and it is considered in the total cost for the non-enhanced solution.

As for the entropy coding method used to encode the shift parameters, following again from the behaviour highlighted in Figure 4.4, the shift parameters can be efficiently compressed using similar methods as the motion vector differences. AVC makes use of variable length coding (VLC) tables that associate each value with codewords. Small codewords are needed to encode small values and progressively longer codewords are used for higher, less probable values. The VLC tables were tuned to efficiently represent the range of values output by the algorithm.

Finally, as the bitstream output by the encoder is modified to accommodate the additional EIP parameters, a suitable decoder needs to be implemented accordingly. The alterations to the decoder are limited to decoding the shift values in all the inter-predicted blocks, and applying the corresponding transformation to compute the enhanced prediction and corresponding reconstruction. As such, while the computation of the shift does add some complexity to the modified encoder (which is anyway limited by the efficiency of the fast algorithm proposed in this chapter), the modifications to the decoder have a

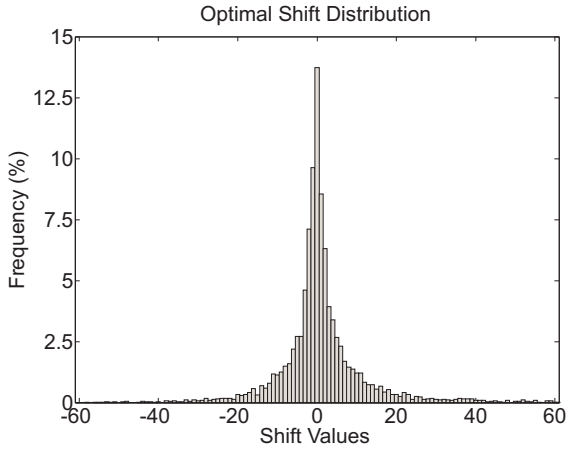



Figure 4.4: Histogram of the optimal shift values for the first 8 frames of the Foreman sequence encoded with $QP = 22$.

negligible impact on the decoder complexity.

## 4.3 EIP and HEVC

### 4.3.1 Background

HEVC makes use of several new techniques and refinements to the inter-prediction module with respect to its counterpart in AVC. Such techniques have a strong impact on the methodology and the performance of the approach presented in this chapter.

First, as shown in Chapter 2, HEVC allows for a recursive partitioning of the current slice in blocks (coding units, CU) at different depths. Depending on the depth, a CU is a square block whose size can be relatively much larger, or conversely much smaller, than the macroblocks of fixed size found in AVC. This flexibility is widely regarded as one of the key factors responsible for the considerably higher efficiency of HEVC than its predecessor: much smaller gains are reported by HEVC over AVC when the CU size is limited to the same macroblock size used in AVC [89].

Allowing the sequence to be partitioned in such a granular way makes the inter-prediction module much more adaptive to local conditions of the signal. This has a negative impact on the EIP due to the fact that HEVC can already compensate for some of the prediction errors that are successfully targeted by the EIP: moreover as the inter-predicted signal is generally more accurate than the one output by AVC, it is also more difficult to enhance.

Another tool peculiar to HEVC which needs to be considered in the context of the EIP is the Merge mode [93], already illustrated in this thesis in subsection 2.4.4. The method consists in reusing the motion information of previously coded blocks for motion compensating the currently encoded block, in order to avoid the additional rates needed to entropy code the motion vector components and reference indexes. The Merge mode is

capable of providing consistent gains during the encoding, based on the assumption that the motion information in spatially or temporally neighbouring blocks may be highly correlated with that of the current block. Such assumption does not hold as strongly in the case of the EIP parameters, which means that using the EIP could consequently reduce the effectiveness of the Merge mode.

Finally, HEVC makes use of another tool which is also potentially disruptive to the effectiveness of EIP referred to as internal bit-depth increase (IBDI) [112]. When this tool is enabled the number of bits used to represent samples in the prediction and reconstructed signals is increased relative to the input bitdepth of the original signal. The higher internal bitdepth than the bitdepth of the input signal allows smaller losses due to a less aggressive truncation of variables during the interpolation of reference frames for the purpose of fractional motion compensation, consequently increasing coding efficiency. The problem when using such a tool along with the EIP is that the shift parameters would similarly require the same level of bitdepth precision which means that the related transmission bitrates would be higher.

As a result of the aforementioned factors, implementing the EIP with shifting transformation in HEVC in the same way as it was integrated in AVC in subsection 4.2.3 does not provide any benefits to the encoding. Some tests were performed on typical video sequences to compare the performances of the EIP with shifting transformation against conventional HEVC. Results of such tests are reported in Table 4-A both in the case when IBDI is disabled or when it is used (with an internal bitdepth set to 10bits). Note that, following the same convention used in the rest of this thesis, positive values of the BD-rate correspond to losses with respect to the anchor. Interestingly these experiment showed that in average 93% of the shift parameters returned by the approach are equal to 0, which indicates that HEVC already compensates for most of the prediction errors targeted by the EIP when used as described in subsection 4.2.3: this is a consequence of the fact that the shifting transformation is not effective in an RD sense in the context of such an high efficient coding scheme. Average BD-rate losses of 0.6% and 0.9% are

obtained respectively when IBDI is disabled or enabled, as an effect of encoding many zero-valued parameters with no corresponding impact on the prediction accuracy.

Some conclusions can be highlighted from these experiments: (i) the EIP with shifting transformation cannot be implemented in HEVC directly within the loop among motion vector candidates; (ii) some of the tools introduced in HEVC need to be adapted to fit the parametric transformations introduced by the proposed approach; (iii) the range of allowed parameters output by the EIP must be carefully adjusted to the standard; and (iv) the EIP may be used to target specific coding conditions instead of targeting improvement of coding efficiency under all conditions.

Two approaches are proposed in the rest of this chapter in the light of these considerations.

### 4.3.2 Merge mode Transformation

The Merge mode can provide considerable gains when tested in HEVC on conventional sequences [93]. While using this tool, the distortion of each Merge candidate is computed making use of the same error metric used for motion estimation (usually the SAD), calculated between the samples in the current prediction unit (PU) and the samples in the inter-prediction block obtained using this Merge candidate. Such distortion is then

Table 4-A: BD-rates of the EIP with shifting transformation against conventional HEVC.

| Resolution | Sequence | FPS | BD-rates (no IBDI) | BD-rates (IBDI 10bits) |
|---|---|---|---|---|
| $832 \times 480$ | Basketball Drill | 30 | 0.8 | 1.5 |
| | Keiba | 30 | 0.4 | 0.5 |
| $416 \times 240$ | RaceHorses | 30 | 1.2 | 0.6 |
| | Mobisode2 | 30 | 0.7 | 0.9 |
| $352 \times 288$ | Foreman | 30 | 1.1 | 1.1 |
| | Crew | 30 | 1.0 | 1.5 |
| | Carphone | 30 | 0.7 | 1.6 |
| | Waterfall | 30 | 0.3 | 0.7 |
| | Silent | 30 | 0.2 | 0.2 |
| | Soccer | 30 | 0.1 | 0.2 |

used to compute an RD cost, and then compared with the cost of the optimal solution output by conventional motion estimation. In case one of the Merge candidates return a smaller RD cost, Merge mode is selected for the current PU.

Very little overhead is required to signal all the information required to compute the prediction in a merged PU, comprised of a flag (to signal whether Merge mode is used in a PU or not) and an index (to extract the correct Merge candidate). For this reason it makes sense to investigate methods with the specific goal of allowing the Merge mode to be used as often as possible. In particular there are several situations in which the prediction extracted using motion information of neighbouring candidates would be sufficiently accurate aside for brightness variations between original and prediction blocks. This happens in many situations in sequences with scene changes or recurrent drastic changes in brightness. The EIP was studied to target such conditions, under the assumption that applying the shifting transformation on each of the Merge candidates might result in a smaller prediction error at an acceptable cost in terms of bitrates.

In particular denote the current PU as $X$, and consider a Merge candidate $k$ with the associated inter-predicted block $P_k$ resulting from using this candidate. The optimal shift parameter $s^o$ can be found for the pair $(X, P_k)$, applying the algorithm in subsection 4.2.1. The corresponding RD cost is modified accordingly as follows:

$$J_k = SAD\left(P_k, X\right) + \lambda_{MRG} R_k + \lambda_s R_s, \tag{4.13}$$

where $\lambda_{MRG}$ and $\lambda_s$ are the Lagrangian multipliers used for computing the RD cost of the Merge index and the shift parameter respectively.

Following the conventional HEVC coding loop, the encoder analyses CUs at different depths, testing for each CU several inter-prediction modes on PUs of different shapes and sizes. Each time the Merge mode is tested, the EIP is used to enhance the predic-

tion returning the transformed inter-predicted block and the optimal parameter for the current PU. If the Merge mode is chosen after RD optimisation, such a parameter is finally encoded in the bit-stream.

An analysis of the behaviour of the EIP parameters in this implementation was performed on some test sequences and is shown here. Experiments were performed on sequences available in the HEVC test set, all with a resolution of $832 \times 480$ samples. Four values of the QP were used, namely 22, 27, 32 and 37 (where QP is incremented by 1 in inter-predicted frames). Note that the internal bitdepth increase was enabled in all these tests and set to an internal bitdepth of 10. This is assumed in all cases in the rest of this chapter unless otherwise specified.

As mentioned in the previous subsection, when applied to the motion estimation loop in HEVC the EIP is not effective because the vast majority of coded PUs result in zero-valued EIP parameters. In order to verify whether similar effects are reported when the EIP is applied to the Merge mode, the average number of zero-valued EIP parameters was computed over the total number of PUs encoded with Merge mode. The results of such experiments are shown in Table 4-B. Tests are presented for each value of the QP and for each CU depth: the EIP was enabled only in PUs extracted from CUs at the corresponding depth, while conventional Merge mode was used in all other cases.

There is clearly a strong correlation between the depth/QP combination and the average number of zero-valued EIP parameters. Smaller blocks (namely those extracted from CUs at depth 3), and/or higher QP values may result in a very high percentage of zero-valued parameters, as high as 74%. Enabling the EIP in these PUs provide no actual enhancements to inter-prediction while at the same time it negatively affects the coding efficiency (due to the relatively high number of bits needed to signal zero valued shift parameters), and as such the approach was disabled in all QP/depth combinations resulting in more than 50% of parameters equal to 0.

Extending this analysis, the histogram of the absolute values of optimal EIP param-

Table 4-B: Percentage of zero-valued EIP parameters in Merge mode.

| QP \ Depth | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 22 | 30.15 | 28.45 | 35.26 | 57.86 |
| 27 | 24.08 | 29.07 | 32.85 | 66.61 |
| 32 | 22.63 | 32.99 | 50.79 | 72.07 |
| 37 | 26.49 | 40.60 | 57.71 | 74.12 |

eters in the case of depth 1, $QP = 37$ is represented in Figure 4.5. The frequency of occurrence of the shift parameters does not follow the same distribution as previously obtained in AVC (in Figure 4.4). In fact excluding the special case when $s = 0$, the frequency of occurrence tends to grow along with increasing values of the parameter up to a certain peak, after which it decreases again. Similar results (at different values of the peak frequency) were obtained for all other QP/depth combinations. This behaviour is an effect of the Lagrangian cost used for selecting the optimal EIP parameter: a very small value of $s$ can only contribute with modest gains in prediction accuracy, while still impacting negatively performances in an RD sense, and therefore it is rarely selected. Conversely higher values have larger benefits on the prediction, up to a certain point where the cost of transmission of $s$ becomes too high for such benefits to have an impact on the coding efficiency

Following from these tests, in order to reduce as much as possible the number of bits required to transmit the shift parameters the EIP was further modified by reducing the number of possible outcomes which this parameter can assume, consequently decreasing the amount of bits required for the related coding. In order to do so a certain quantisation

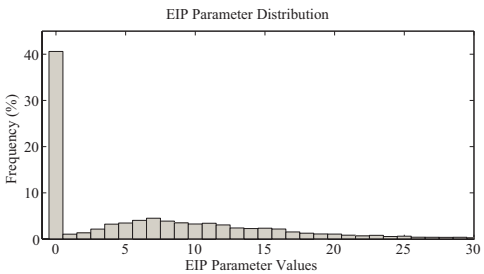



Figure 4.5: Histogram of the EIP parameters in the Merge mode.

step is considered when computing and encoding the parameter $s$. It is easy to show that the SAD as a function of the shift parameter is a convex function, therefore, the optimal quantised parameter in terms of SAD is this value at the given quantisation step that is nearest to the optimal non-quantised parameter. Formally given a certain quantisation step $QS$ and the optimal shift parameter $s^0$, this corresponds to:

$$s^o_{QS} = round\left(\frac{s^o}{QS}\right) QS, \tag{4.14}$$

where the $round(\bullet)$ operator approximates to the nearest integer. While convexity is not formally true when considering the RD cost, still a similar behaviour can be expected and it is assumed in the rest of this chapter.

The problem becomes then that of effectively deriving $QS$ to represent the range of EIP parameters while at the same time limiting the related bitrates. Such derivation is performed here again separately for each QP/depth combination. The same data used to derive Table 4-B was used. In particular, denote as $N_{s,QP,depth}$ the number of EIP parameters equal to $s$ output while coding these test sequences under a certain QP/depth combination. Assume that a maximum absolute value of the EIP parameter $s_{max}$ is output. Then the prediction error produced as an effect of using this quantisation can be estimated as:

$$E(QS)_{QP,depth} = \sum_{s=0}^{s_{max}} N_{s,QP,depth}|s - s_{QS}|. \tag{4.15}$$

Conversely, for each value of $QS$ under a certain QP/depth combination, an estimate of the number of bits saved as an effect of quantising the parameters with a step $QS$ can be computed by using variable length coding tables. Denote this number as $B(QS)_{QP,depth}$.

The error $E(QS)_{QP,depth}$ and rate saving $B(QS)_{QP,depth}$ were computed for all values

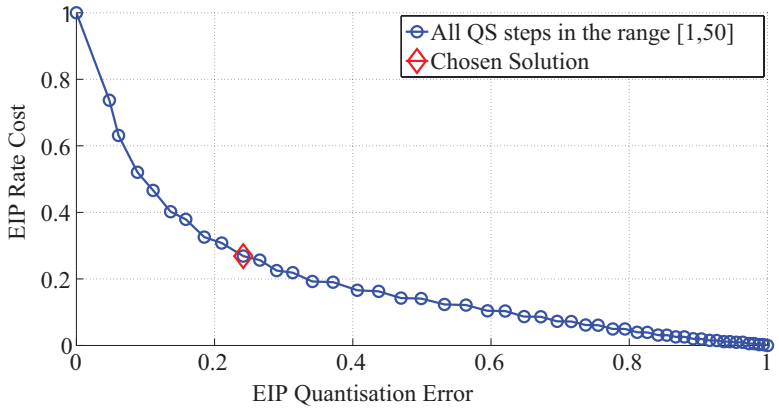



Figure 4.6: EIP Quantisation Error vs EIP Rate Cost.

of $QS$ within a predetermined range, for each considered QP/depth combination. These are shown in Figure 4.6 in the case of depth equal to 0, $QP = 37$. Finally the step $QS_{QP,depth}$ was selected as the value that returns the point in the plot which is nearest to the origin in terms of Euclidean distance. The process was repeated for each QP/depth combination to obtain a matrix of optimal quantisation step values as shown in Table 4-C. Note that the quantisation step is not computed in case the EIP is disabled in this QP/depth combination.

Table 4-C: Optimal quantisation steps for each QP/depth combination.

| | | Depth | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 |
| QP | 22 | 4 | 4 | 6 | N/A |
| | 27 | 4 | 6 | 8 | N/A |
| | 32 | 4 | 8 | N/A | N/A |
| | 37 | 6 | N/A | N/A | N/A |

Finally the encoder was modified to make use of these quantisation steps. Four ranges are considered for the QP, centred respectively in 22, 27, 32 and 37. The encoder classifies each PU according to such ranges and the depth of the CU, and extracts the corresponding value of $QS$. When Merge mode is tested, the EIP is applied to the prediction candidates. The optimal $s$ is first computed and successively quantised using $QS$. The proposed approach achieves consistent gains with respect to conventional HEVC as shown in the rest of this chapter.

### 4.3.3 Mode Decision using EIP

In the previous subsection it was shown that the EIP with shifting transformation can be successfully applied in the context of the Merge mode included in HEVC following some adaptation to the new characteristics of the standard. In particular the optimal shift parameters are quantised depending on local conditions of each block to reduce the related bitrates. The Merge mode is a relatively small component of a typical HEVC encoder and as such it is reasonable to expect a relatively modest impact of the EIP on the total coding performance. Still, the approach proposed in the previous subsection highlights the fact that under certain conditions the shifting transformation can be successfully applied to HEVC to enhance the prediction accuracy. Therefore it makes sense to investigate the application of the EIP under similar conditions, in the context of other modules of the encoder which have a bigger impact on the overall coding efficiency.

An important component of HEVC is the mode decision module responsible for selecting the optimal inter-prediction mode for a given CU. Mode decision is typically used in HEVC at the encoder side after motion estimation is performed on all the prediction blocks within the CU. Applying the EIP during this step requires less parameters to be encoded (i.e. a single parameter per CU) than if it is applied directly during motion estimation (in which case a parameter is needed for each prediction block). While this approach does not allow the EIP to directly change the outcomes of motion estimation, it can still influence coding efficiency by enhancing the entire inter-predicted CU.

Due to the relatively large number of motion vector candidates that need to be tested during the motion estimation loop, the encoder makes use of an estimated RD cost computed using the SAD between prediction block and original signal. On the contrary only a few different inter-prediction modes are available to be selected (i.e. 8 modes for each depth in the case of HEVC) which means that the mode decision module can perform this selection using a more accurate RD cost, without significant impacts on the coding complexity. In particular typical codecs compute for the entire

CU the residual samples $R = P - X$ obtained by means of each of the inter-prediction modes. These are transformed, quantised and entropy coded together with the motion information, to obtain an exact measure of the rate needed for a given inter-prediction mode. Finally quantised coefficients are dequantised, inverse transformed and added back to the prediction to obtain the reconstructed block $X_{rec}$. This is then used to compute the distortion against the original signal (typically by means of SSD). A RD cost is defined accordingly, and finally the encoder selects the optimal inter-prediction mode as the one at minimum cost.

Integrating the EIP with shifting transformation in such an architecture is not trivial: the algorithm in subsection 4.2.1 was defined taking into consideration the prediction error $D_{pr} = \Delta(P, X)$ and cannot be easily adapted to the case of $D_{rec} = \Delta(X_{rec}, X)$. On the other hand, a brute-force approach where a set of values of $s$ are used to compute the corresponding reconstructed block $X_{rec}$ and then compared in an RD sense is also not optimal, due to the extremely high related computational costs needed for the forward and inverse transform operations.

Consider though that a certain parameter $s$ is used to transform all prediction blocks within the current CU, obtaining the block $P_s$, and that this block is then used to obtain the residual samples $R_s$. Consider that these are transformed, quantised and encoded to the bitstream, and finally decoded obtaining the block $R_{rec,s}$; this is used to compute the reconstructed block $X_{rec,s}$. Denote the CU size as $B = M \times N$ where $M$ and $N$ are the CU height and width respectively. Denote as $p_s(i)$, $r(i)$, $r_{rec,s}(i)$ and $x_{rec,s}(i)$ the samples in $P_s$, $R_s$, $R_{rec,s}$ and $X_{rec,s}$ respectively where $i = 0, ..., B - 1$. Then the

corresponding reconstruction error using SSD is:

$$\begin{aligned} D_{rec,s} &= \sum_{i=0}^{B-1} \left[x_{rec,s}\left(i\right) - x\left(i\right)\right]^2 = \sum_{i=0}^{B-1} \left[r_{rec,s}\left(i\right) + p_s\left(i\right) - x\left(i\right)\right]^2 = \\ &= \sum_{i=0}^{B-1} \left[p_s\left(i\right) - x\left(i\right)\right]^2 + 2\sum_{i=0}^{B-1} \left[p_s\left(i\right) - x\left(i\right)\right] r_{rec,s}\left(i\right) + \sum_{i=0}^{B-1} \left[r_{rec,s}\left(i\right)\right]^2 = \\ &= D_{pr,s} - 2\sum_{i=0}^{B-1} r\left(i\right) r_{rec,s}\left(i\right) + \sum_{i=0}^{B-1} \left[r_{rec,s}\left(i\right)\right]^2. \end{aligned} \tag{4.16}$$

Assuming that the forward and inverse transform and quantisation processes do not change the sign of the elements in $R_{rec}$ with respect to the original elements in $R$, or:

$$\sum_{i=0}^{B-1} r\left(i\right) r_{rec,s}\left(i\right) \geq 0,$$

and under the assumption that the quantisation error is signal dependent, which means that the energy of the reconstructed residuals $\sum_{i=0}^{B-1} \left[r_{rec,s}\left(i\right)\right]^2$ depends on the energy of the non-quantised residuals $D_{pr,s}$, then it can be assumed that minimising $D_{pr,s}$ also minimises $D_{rec,s}$ with respect to $s$. The following can be easily obtained:

$$D_{pr,s} = Bs^2 + 2\left[\sum_{i=0}^{B-1} \left[x(i) - p(i)\right]\right] s + \sum_{i=0}^{B-1} \left[x(i) - p(i)\right]^2. \tag{4.17}$$

The optimal parameter $s^o$ can be correspondingly computed as:

$$s^o = \frac{1}{B}\left[\sum_{i=0}^{B-1} x\left(i\right) - \sum_{i=0}^{B-1} p\left(i\right)\right]. \tag{4.18}$$

In order to use the EIP, the parameter needs to be transmitted along with the motion information for each CU. While the actual entropy encoding of this parameter may

be integrated within the CABAC architecture included in HEVC, the rate required to transmit such parameters can be estimated assuming that variable-length coding (VLC) tables are used instead. This allows for a numerical measure of this rate that does not depend on statistics of the current signal being encoded. In particular given a parameter $s$, VLC tables need $b(s)$ bits to encode $s$ where:

$$b(s) = 1 + 2\lfloor log_2\left(2|s| + 1\right)\rfloor, \tag{4.19}$$

and where the $\lfloor\bullet\rfloor$ operator results in the highest integer number lower than its argument. Note that at least one additional bit needs to be transmitted for each CU, even when the EIP is not used, to signal a parameter $s^o = 0$.

Following again from the assumption that the shift parameter can be quantised to improve the compression efficiency of the approach, the problem is then that of defining appropriate quantisation steps in the context of this new implementation of the EIP module. The optimal parameter $s^o$ quantised with a step $QS$ becomes equal to $s^o_{QS} = round\left(\frac{s^o}{QS}\right) QS$. Using the expression in Equation 4.19, coding this parameter requires a number of bits equal to:

$$b(s^o_{QS}) = 1 + 2\left\lfloor log_2\left(2\left|\frac{s^o}{QS}\right| + 1\right)\right\rfloor. \tag{4.20}$$

While obviously higher values of $QS$ require fewer bits for the encoding, the prediction efficiency decreases due to using an EIP parameter different from the optimal $s^o$. At this purpose define the function $\Phi(s) = D_{pr,s} - D_{pr,s^o}$ as the difference between the prediction distortion using a parameter $s$ and the minimum distortion using the parameter $s^o$. Recall that Equation 4.17 presents a closed form solution for the prediction distortion obtained using a certain parameter $s$. Define also as $\varphi = \Phi(0) = D_{pr,0} - D_{pr,s^o}$ the difference between the non-enhanced distortion (i.e. with $s = 0$) and the enhanced

distortion. Notice that $D_{pr,s}$ in Equation 4.17 is an even function of $s$ with a single minimum in $s^o$, as shown in Fig. 4.7. Hence $\Phi(s)$ is also even with the same single root $s^o$ or: $\Phi(s) = \alpha\,(s - s^o)^2$. But $\Phi(0) = \alpha(s^o)^2 = \varphi$, which means:

$$\Phi(s) = \frac{\varphi}{s_0^2}\,(s - s^o)^2\,. \tag{4.21}$$

The RD cost of the optimal solution $s^o$ can be defined using Lagrangian optimisation as:

$$\begin{aligned} J_{pr}(s_0) &= D_{pr,s^o} + \lambda_{pr} b_{MI} + \lambda_{pr} b\,(s^o) \\ &= D_{pr,0} - \varphi + \lambda_{pr} b_{MI} + \lambda_{pr} b\,(s^o)\,, \end{aligned} \tag{4.22}$$

where $b_{MI}$ is the cost of transmission of the motion information, and $\lambda_{pr}$ is the corresponding Lagrangian multiplier as defined in the standard.

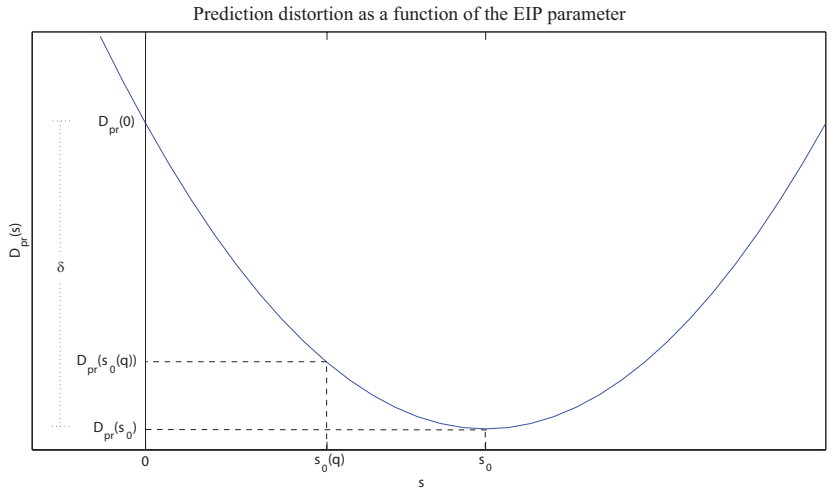



Figure 4.7: Behaviour of the prediction distortion in terms of optimal and quantised EIP parameters.

Consider now the prediction RD cost obtained using the quantised parameter $s^o_{QS}$. In order for the parameter to be effective for a particular CU, the condition $J_{pr}(s^o_{QS}) < J_{pr}(0)$ should hold for that CU. By means of simple substitutions this condition corresponds to:

$$D_{pr,0} - D_{pr,s^o_{QS}} > \lambda_{pr} b(s^o_{QS}). \tag{4.23}$$

And finally following from Equations 4.20 and 4.21:

$$\frac{\varphi}{(s^o)^2} s^o_{QS} \left(2s^o - s^o_{QS}\right) > \lambda_{pr} + 2\lambda_{pr} \left\lfloor log_2 \left(2 \left| \frac{s^o}{QS} \right| + 1\right) \right\rfloor, \tag{4.24}$$

where the aforementioned expression depends, for a given value of the quantisation step $QS$, only on the parameters $s^o$ and $\varphi$.

The condition in Equation 4.24 can be used to select the optimal quantisation step under different optimisation targets. For instance, a target might consist in choosing the values of $QS$ for each QP/depth combination to maximise the number of $(\varphi, s^o)$ pairs where the condition is satisfied. The quantisation steps obtained with this criterion (which is referred to here as Utilization target) would guarantee that the EIP is used in the highest number of CUs. Such target does not take into account the actual impact of the quantisation on the prediction RD cost. For this purpose a different approach can be used where the quantisation steps are instead selected as the values that maximise the impact of the EIP, that is the difference between the enhanced and non-enhanced prediction RD costs. Formally, this corresponds to the value of $QS$ that make the cumulative difference $\left(J_{pr}(s^o_{QS}) - J_{pr}(0)\right)$ the largest possible (among all considered values of $QS$) on all $(\varphi, s^o)$ pairs that satisfy Equation 4.24. Such target is referred to

here as Impact target.

Both targets were tested. In particular, tests were performed on a selection of test sequences, at the same 4 QP values as used previouslty in this chapter. In each test the EIP is enabled only on CUs at a particular depth. For each CU when the EIP is enabled, the pair $(\varphi, s^o)$ is considered. The resulting pairs were used to compute the optimal quantisation steps for the Utilisation and Impact target, as shown in Table 4-D and Table 4-E respectively.

The EIP is then used as illustrated in this subsection, and quantised depending on the correct quantisation step as extracted from the considered table. The EIP parameter at minimum cost is used for enhancing inter-prediction of the entire CU, and it is encoded and transmitted in the CU header.

Table 4-D: QS values found using the Utilization target.

| | | Depth | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 |
| QP | 22 | 1 | 1 | 2 | 3 |
| | 27 | 1 | 1 | 2 | 4 |
| | 32 | 1 | 1 | 2 | 4 |
| | 37 | 1 | 1 | 2 | 4 |

Table 4-E: QS values found using the Impact target.

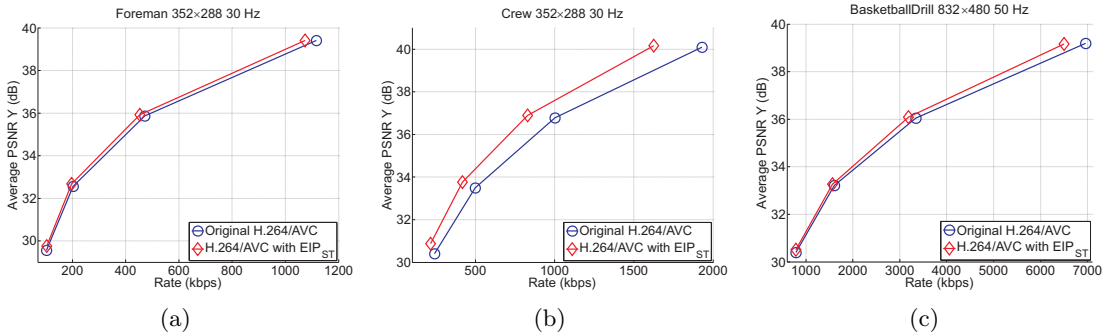
| | | Depth | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 |
| QP | 22 | 1 | 1 | 3 | 7 |
| | 27 | 1 | 1 | 3 | 7 |
| | 32 | 1 | 1 | 3 | 7 |
| | 37 | 1 | 3 | 3 | 7 |

## 4.4 Results

### 4.4.1 EIP in AVC

The EIP with shifting transformation was implemented and tested in the context of the AVC using the JM reference software version 18.2 [72]. The approach was tested on several popular sequences at different resolutions. In all cases, the full length of the sequences is used (e.g. 300 frames are tested for sequences at 30 Hz). CAVLC was used for the entropy coding of the coefficients (more information on CAVLC can be found in Appendix A). Four QP values are used in the tests: 22, 27, 32 and 37 for intra-predicted frames, and 23, 28, 33 and 38 for inter-predicted frames respectively.

Selected results are shown in Figure 4.8 in terms of RD curves against the non-enhanced conventional AVC encoder. Three curves are shown, for the Foreman and Crew sequences at CIF resolution and for the Basketball Drill sequence at $832 \times 480$ resolution, all at a framerate of $30Hz$. The EIP with shifting transformation outperforms the non-enhanced encoder in all these tests, with gains that are generally distributed towards high reductions in the bitrate and unchanged or slightly improved reconstruction PSNRs, meaning that same quality is obtained after the encoding but at the cost of smaller bitrates. In one case (Crew), high gains in the reconstruction PSNR are also obtained.




(a) (b) (c)

Figure 4.8: Rate-distortion curves for: (a) Foreman $352 \times 288$ 30 Hz; (b) Crew $352 \times 288$ 30 Hz; and (c) Basketball Drill $832 \times 488$ 50 Hz.

A frame-by-frame breakdown of a typical case is shown in Figure 4.9 for the first 100

frames of the Foreman sequence encoded with $QP = 22$. The plot at the top of the figure shows the difference between the number of bits required to encode each frame with conventional AVC and proposed encoder. The approach requires less bits to encode all but one frame in the sequence, consistently providing lower bitrates. On the other hand, the difference in the reconstruction PSNR for each frame between the proposed modified encoder and the original H.264 is shown in the plot at the bottom of the figure. Negligible decreases in PSNR are obtained in the majority of the frames: the PSNR difference is higher than $-0.06$ in 81% of the cases, and it is higher than $-0.08$ in 92% of the cases.

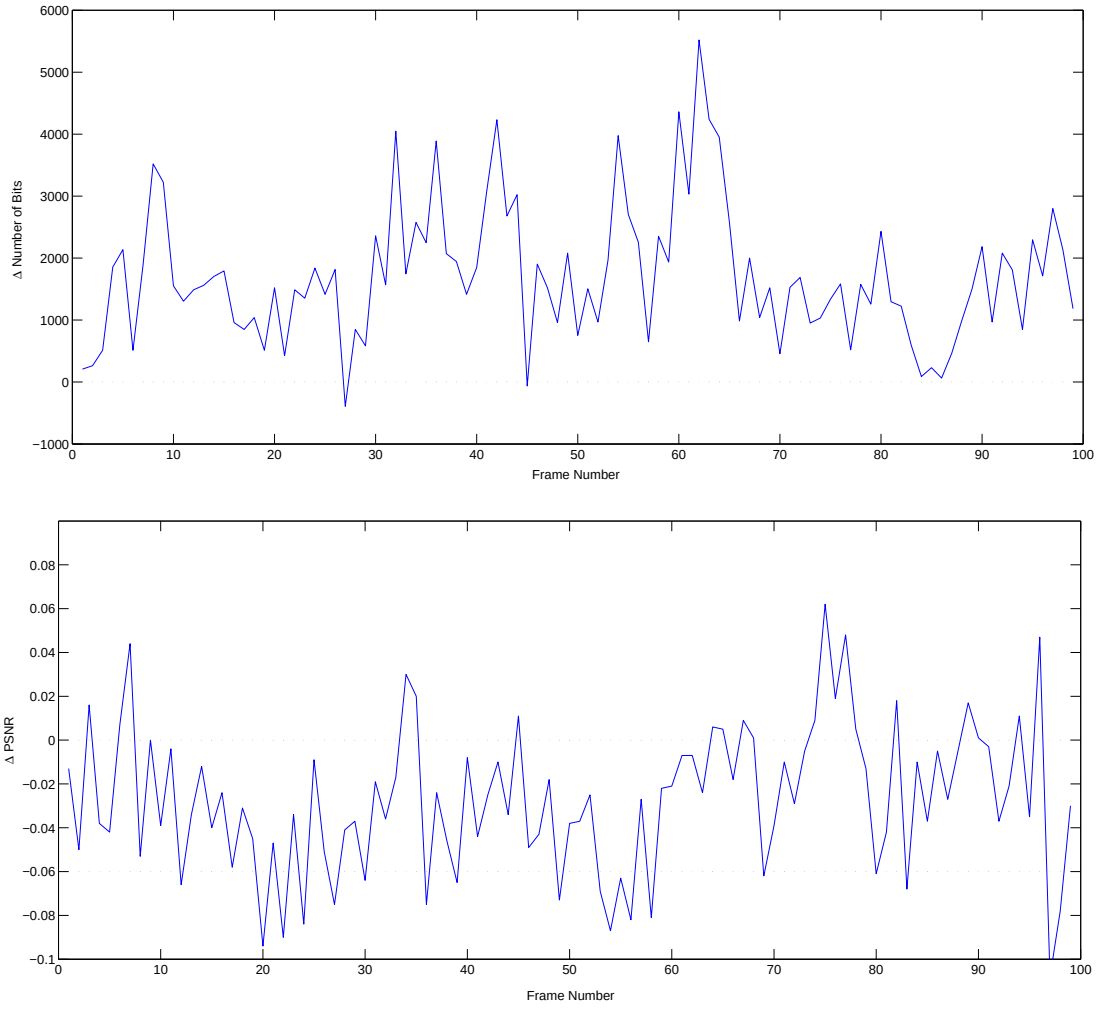



Figure 4.9: Bitrate and PSNR difference between conventional AVC and AVC with EIP for the Foreman sequence with $QP = 22$.

Full results of the proposed method are shown in Table 4-F. For completness, these are compared with the weighted prediction (WP) tool as it is available in the AVC standard. The explicit mode was used for these tests as it is the tool more similar to the proposed method. The results are presented in terms of BD-rates, where the original AVC encoder (with no WP) is used as anchor for all tests. The method used to compute the WP parameters is LMS (least mean squares) with chroma support disabled and weighted references used for motion estimation in the encoder configuration [77].

For most sequences, the gain achieved using the proposed encoder is between $-4\%$ to $-6\%$, again mostly distributed towards high reductions in the bitrate and almost unchanged PSNRs. The proposed approach outperforms the original AVC in all sequences but Mobile at 7.5 Hz, where it increases the bitrate by an average of 0.6%. Some sequences result in lower gains ($-2.3\%$ gains for Party Scene, and $-0.3\%$ for Mobile 30 Hz). For two sequences, Crew and Waterfall, the approach shows significantly large gains (up to $-19\%$ for Crew and $-11\%$ for Waterfall).

On the other hand, the WP is mostly ineffective when used on the same sequences, providing unchanged or slightly worse rates on almost all the tests. This is due to the fact that WP was developed for particular situations such as scene transitions or fades which uniformly affect the entire frame. In all other cases the explicit mode actually deteriorates the encoder performance mostly due to the transmission of the parameters which is not compensated by any gain in the prediction accuracy.

Note in fact that the that best results for the WP were obtained in the Waterfall sequence at 30 Hz, where it provided a modest $-0.2\%$ BD-rate reduction. Again this is a consequence of very large gains in some sparse frames and almost no effect in the remaining frames. The EIP with shifting transformation instead produced consistent results among all the frames in the tested sequences, providing some bitrate reduction and/or quality improvement in the vast majority of the cases.

Finally, the Structural Similarity index (SSIM) [14] was also used as an alternative

measure to the PSNR to validate the claim that, despite the proposed encoder achieves typical lower bitrates than conventional AVC, the overall quality obtained using the two encoders is the same. SSIM values returned by the two encoders are identical (up to the third decimal value); for instance average values obtained for the Crew sequence are 0.96 at $QP = 22$ and 0.78 at $QP = 37$, exactly the same as those obtained using a conventional AVC encoder.

### 4.4.2 The EIP in HEVC

First, the EIP with shifting transformation was integrated in the context of the Merge mode as used in HEVC. The approach in subsection 4.3.2 was implemented in the context of the HM reference software version HM 12.0 [85] under the the low-delay configuration [113]. Internal bitdepth increase was set to 10 in all cases. Tests were performed on several popular test sequences. Three sequences at different resolutions and framerates are of particular interest to the approach, namely Crew ($352 \times 288$, 30 Hz), ElephantsDream

Table 4-F: Results of proposed approach against conventional AVC.

| Resolution | Sequence | FPS | BD-rate (%) | |
|---|---|---|---|---|
| | | | EIP | WP |
| $832 \times 480$ | PartyScene | 50 | $-2.4$ | $+0.0$ |
| | BasketBall Drill | 50 | $-4.8$ | $+0.0$ |
| | RaceHorses | 30 | $-4.1$ | $-0.0$ |
| $352 \times 288$ | Carphone | 30 | $-6.0$ | $-0.0$ |
| | Foreman | 30 | $-4.7$ | $+0.0$ |
| | Crew | 30 | $-19.5$ | $-0.1$ |
| | Mobile | 30 | $-0.3$ | $-0.0$ |
| | Mother-Daughter | 30 | $-4.0$ | $+0.0$ |
| | Bowing | 30 | $-4.2$ | $+0.0$ |
| | Waterfall | 30 | $-11.6$ | $-0.1$ |
| | Carphone | 7.5 | $-5.3$ | $+0.0$ |
| | Foreman | 7.5 | $-4.6$ | $+0.1$ |
| | Crew | 7.5 | $-13.3$ | $-0.0$ |
| | Mobile | 7.5 | $+0.6$ | $+0.0$ |
| | Mother-Daughter | 7.5 | $-5.0$ | $+0.0$ |
| | Bowing | 7.5 | $-10.1$ | $+0.1$ |
| | Waterfall | 7.5 | $-9.9$ | $-0.2$ |

($352 \times 288$, 24 Hz) and Mobisode2 ($416 \times 240$, 30 Hz). These sequences all present local variations of the brightness in neighbouring frames, which means the EIP is particularly suited for the encoding. For instance, two consecutive frames of the Mobisode2 sequence are shown in Figure 4.10. Major local brightness variations between the two frames are delimited by dashed lines in the figure.

Full results are presented in Table 4-G. The modified encoder always outperforms conventional HEVC in all tests. The three aforementioned sequences are highlighted in bold in the table. Clearly these correspond to the cases in which the EIP has the largest impact on the coding efficiency when integrated in the context of the Merge mode.

Despite the limited impact of the method, which only affects a relatively small component in the HEVC coding loop, still the approach is capable of providing consistent coding gains in terms of coding efficiency. Up to $-2.6\%$ BD-rate gains are obtained in the case of the Mobisode2 sequence, and in average $-0.6\%$ gains in all tested sequences. Interestingly while the approach provides best performance when coding particular content, still it provides some gains in all cases and never perform worse than conventional HEVC.

The approach is also capable of increasing the number of times that the merge mode is selected. In conventional HEVC, the Merge mode is reportedly used in average in around 15% of the prediction blocks [93]. This number was verified while performing

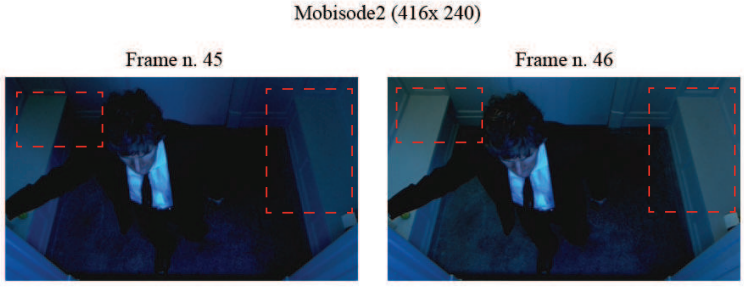



Figure 4.10: Consecutive frames in the Mobisode2 sequence. Areas with major local brighness variations are delimited by dashed lines.

these experiments: for instance for $QP = 32$, 22% of the inter-predicted blocks were coded using the Merge mode in the Mobisode2 sequence, and 12% in the PartyScene sequence. When using EIP such numbers increase to 27% and 15% respectively. The approach is successful in increasing the chances of selection of Merge mode for a given block.

As for complexity of the approach, note that the method requires very limited amount of additional calculations, mainly to compute the optimal EIP parameter for each Merge candidate. As a result, the modified encoder requires only in average around 5% additional time to complete the encoding, while the approach has negligible effects on the decoding time (in average less than 1% increase).

Then the EIP was integrated in the context of HEVC mode decision as illustrated in subsection 4.3.3. Test conditions for these experiments are identical to those used for the experiments in the previous subsection. Similar to previous experiments, also in this case best results can be expected for the Mobisode2, Crew and ElephantsDream sequences due to their characteristics which make them particularly suited to the EIP with shifting transformation.

Results are presented in Table 4-H.

Gains of the method are distributed towards reduction in the bitrates at unaffected

Table 4-G: BD rate of the EIP in Merge mode against conventional HEVC.

| Resolution | Sequence | FPS | BD-rates |
|---|---|---|---|
| 416 × 240 | **Mobisode2** | 30 | **-2.6** |
| 352 × 288 | **Crew** | 30 | **-0.7** |
| | **ElephantDream** | 24 | **-1.0** |
| 832 × 480 | BQMall | 30 | −0.2 |
| | Keiba | 30 | −0.1 |
| | PartyScene | 50 | −0.3 |
| | RaceHorses | 30 | −0.2 |
| 416 × 240 | BlowingBubbles | 50 | −0.4 |
| 352 × 288 | Carphone | 30 | −0.1 |
| | Foreman | 30 | −0.2 |
| | Mobile | 30 | −0.3 |

quality of the decoded signals. This statement was validated using the Structural Similarity index (SSIM), as shown in Table 4-I for the three sequences which are mostly affected by the method. The decoded signals output by the approach are visually indistinguishable to those output by conventional HEVC.

## 4.5 Conclusions

Conventional video coding methods may use some techniques based on reference frame transformation, such as global motion estimation or weighted prediction. These approaches

Table 4-H: BD rate of the EIP in mode decision against conventional HEVC.

| Resolution | Sequence | FPS | BD-rates |
|---|---|---|---|
| 416 × 240 | **Mobisode2** | 30 | **-2.3** |
| 352 × 288 | **Crew** | 30 | **-1.0** |
| | **ElephantDream** | 24 | **-2.2** |
| 832 × 480 | BQMall | 30 | −0.4 |
| | Keiba | 30 | −0.3 |
| | PartyScene | 50 | +0.0 |
| | RaceHorses | 30 | −0.3 |
| 416 × 240 | BlowingBubbles | 50 | −0.5 |
| 352 × 288 | Carphone | 30 | −0.2 |
| | Foreman | 30 | +0.1 |
| | Mobile | 30 | −0.4 |

Table 4-I: SSIM indexes of proposed approach and conventional HEVC.

| Sequence | QP | SSIM (HEVC) | SSIM (EIP) |
|---|---|---|---|
| Mobisode2 | 22 | 0.986 | 0.987 |
| | 27 | 0.980 | 0.980 |
| | 32 | 0.971 | 0.971 |
| | 37 | 0.951 | 0.950 |
| Crew | 22 | 0.979 | 0.979 |
| | 27 | 0.958 | 0.958 |
| | 32 | 0.922 | 0.923 |
| | 37 | 0.873 | 0.874 |
| ElephantsDream | 22 | 0.927 | 0.925 |
| | 27 | 0.886 | 0.886 |
| | 32 | 0.800 | 0.801 |
| | 37 | 0.738 | 0.738 |

generally suffer from strong limitations, in that the related transformations take only into consideration global characteristics of the frame and do not depend on local content. A new approach to enhance the inter-prediction module in typical video coding schemes was presented in this chapter, based on the assumption that inter-prediction can be enhanced by complementing motion compensation with a completely independent additional tool. Such tool, referred to as Enhanced Inter-prediction (EIP) acts at the same time as motion estimation, enhancing each prediction candidate with the goal of returning a new solution which is better suited for the local content currently being encoded.

The EIP was validated using the simple but successful shifting transformation, and it was studied in the context of current and next generation video coding standards. In particular, an algorithm to find the optimal EIP parameter and related distortion savings in a single step was proposed in this chapter. The algorithm was used to implement the EIP in the context of the AVC standard. The method was integrated within existing RD schemes, and extensively tested resulting in consistent gains with respect to conventional AVC coding.

The EIP was also studied in the context of the next generation HEVC standard. A similar approach as the one used in AVC was found to be not optimal for the new standard. This is mostly due to the already very high efficiency of HEVC, and also to limitations of the approach in the context of some of the new tools introduced in the standard. Therefore, methods to tackle such limitations in order to apply EIP to HEVC and target specific coding conditions were then studied. The HEVC was implemented in the context of the Merge mode and mode decision modules in the standard. In both cases, the method was shown capable of outperforming conventional HEVC in the majority of the tests.

# Chapter 5

# Transform Domain Prediction for High Quality Video Coding

High quality video coding opens challenging problems which cannot be solved using conventional technology. An analysis of HEVC intra-prediction under high quality constraints is presented in this chapter, showing that conventional schemes may not provide sufficient prediction accuracy particularly at certain frequency components of the signal. The analysis also shows that such issues typically arise in a predictable way depending on local characteristics of the portion of the signal currently being encoded. Novel methods which combine prediction in the spatial and transform domain are presented in the chapter to target these issues.

## 5.1 High Quality Video Coding

### 5.1.1 Background

Video coding standards are mostly designed for efficient usage for mass scale distribution. Typical prediction schemes included in such standards target the delivery of very high

compression ratios, at the cost of a degradation of the decoded signal to medium or low quality.

While such levels of quality are acceptable for some purposes, there are many applications in which higher levels of quality are necessary. In these cases it is crucial that the decoded video signal is visually as similar as possible to the original signal prior to the encoding. Note that, while many definitions would be possible, high quality is defined in this thesis in terms of the QP values used during the encoding, as will be detailed in the rest of this chapter. Under these constraints it is difficult to predict the fine granularity details of the signal needed to preserve such levels of quality. Consequently, even the most advanced compression schemes are less efficient and deliver high bitrates. As a result the efficiency of HEVC decreases becoming closer to that of its predecessor AVC, providing only minor improvements as it was recently demonstrated via experimental validation [114]. Similarly, when coding still images at such levels of quality, HEVC results in considerably less improvements compared with JPEG2000 [88]. Conventional prediction methods rely on spatial interpolation of neighbouring samples, which typically results in a soft, blurred prediction signal. Such signals are suboptimal for high quality coding as they are not capable of efficiently delivering the high frequency content.

Consequently, novel video coding schemes are being investigated specifically at this purpose. In particular at the time of writing one of the extensions to the HEVC standard, referred to as Range Extensions [115], is currently being developed to also include tools for lossless video coding. As a response to the development of such extensions, many methods have been proposed for improving the efficiency of lossless HEVC. Most of these methods are based on the idea that, due to the fact that no transform and quantisation are used in the coding scheme, it makes sense to eschew the classic concept of block-based prediction for a more granular prediction at a sample level [116]. Residual differential pulse-code modulation [117], sample-based angular prediction [118], sample-based weighted prediction [119] or the pixel-based averaging predictor [120] are all examples of methods that have been recently proposed based on such an idea.

Unfortunately, lossless video coding typically requires extremely high bitrates, which may not be acceptable for some of these applications. Typical examples of this kind of high quality applications can be found in medical imaging services, screenshot sharing tools, or screen mirroring systems when the content on the screen of a device is mirrored in real-time to a different device. Moreover, with the increasing demand for ultra high-definition televisions capable of displaying content at very high framerates and high bitdepths, delivering good quality of the decoded video sequences is becoming an extremely important issue even in the context of consumer applications. Users expect video content at as good quality as possible, with the lowest visible coding distortion, while still achieving acceptable compression ratios to allow for cost-effective content delivery. Mathematically lossless coding is not a viable solution and for this reason other methods should be investigated which still rely on block-based compression schemes. At the time of writing, further extensions to HEVC with the goal of addressing these applications are currently under development [121] [122].

At this purpose, an analysis of conventional intra-prediction methods as used within the HEVC standard is presented in this chapter, with the goal of evaluating the performance of each intra-prediction mode on predicting different frequency components of the original signal under high quality constraints. Note that the conventional HEVC scheme would not allow for such an analysis because the prediction and original signals are subtracted one from the other in the spatial domain, and only the residual signal is subsequently transformed to transform domain. In order to allow the analysis, modified encoder and decoder schemes making use of direct transformation of the prediction blocks are first introduced, as the essential base for the analysis and methodology presented in this chapter.

### 5.1.2 Direct Transformation of Prediction Blocks

Consider that a certain $N \times N$ square block of samples $X$ is being encoded using intra-prediction. Consider also that an equally sized block of samples $P$ is being considered

as a prediction for $X$, obtained from one of the possible intra-prediction modes. Note that only square blocks are considered for intra-prediction in this chapter, due to the fact that the analysis and proposed methods are integrated within HEVC (which only allows for square intra-predicted blocks). The methodology may be easily extended to rectangular blocks.

In conventional video codecs the residual samples $R$ are computed in the spatial domain from the original pixels $X$ and a prediction $P$ as $R = X - P$. The block of residuals is then transformed to the transform domain. Either the block is split in smaller blocks which are independently transformed, or the full block is processed as a whole. For simplicity, without loss of generality only this second case is considered here. The transform is performed by means of an $N \times N$ transform base matrix $Q$ as in:

$$\widetilde{R} = QRQ^{\top}.$$

The transformed samples are successively quantised to obtain the coefficients $\widetilde{C}$. These steps are illustrated in the scheme in Figure 5.1 (a). At the decoder side, the coefficients are extracted from the bitstream, dequantised (i.e. rescaled), and inverse transformed. Due to the fact that the quantisation is a non-reversible operation, the block $R_{dec}$ is different than the block of residuals $R$. $R_{dec}$ is added to the prediction $P$ in the spatial domain to obtain the reconstructed block $X_{dec}$, as in the scheme in Figure 5.1 (c).

In this chapter, different encoder and decoder schemes are considered as follows. The prediction and original signals are directly transformed to the transform domain, to obtain respectively $\widetilde{P}$ and $\widetilde{X}$, as illustrated in Figure 5.1 (c). These are used to obtain the residuals $\widetilde{R}$, which are then quantised as in conventional coding. At the decoder side the prediction block is transformed to the transform domain to obtain $\widetilde{P}$, and the coefficients $\widetilde{C}$ are dequantised to obtain $\widetilde{R}_{dec}$. These are summed to obtain the reconstructed block $\widetilde{X}_{dec}$ in the transform domain, which is finally inverse transformed to return $X_{dec}$, as in

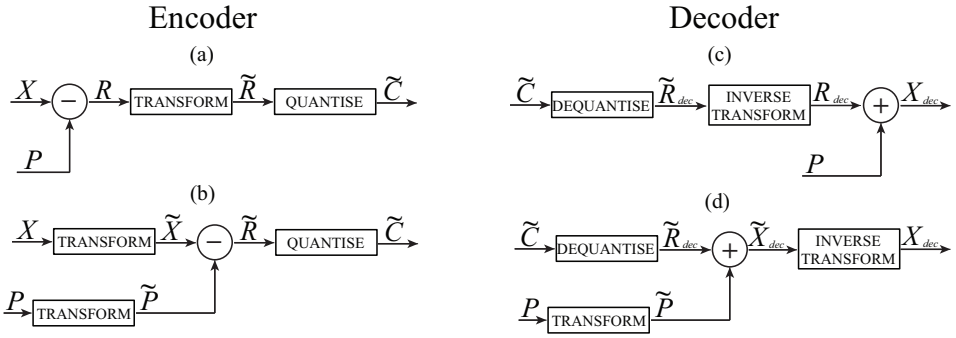



Figure 5.1: Conventional encoder (a) and decoder (c) schemes compared with the proposed encoder (b) and decoder (d) schemes with direct transformation of prediction.

the scheme in Figure 5.1 (d).

Note that similar schemes have already been used in video coding, though they have been applied with different purposes and in different modules of encoder and decoder. A method was proposed [123] in which motion compensated prediction and original signals are separately transformed. In this method the inter-predicted samples are transformed and scaled using pre-computed weights, before calculating the residuals directly in the transform domain. The weights are fixed on a sequence basis, and are transmitted in the bitstream to be used at the decoder side. The method was further extended [124] to include a recursive calculation of the weights avoiding the need for additional side information.

If the same $X$ and $P$ are used as input to the two schemes in Figure 5.1 (a) and (b), exactly the same residual $\widetilde{R}$ should be obtained in the transform domain. The linearity of the transform is easily shown as follows:

$$\widetilde{R} = \widetilde{X} - \widetilde{P} = \left(QXQ^{\top} - QPQ^{\top}\right) = Q\left(X - P\right)Q^{\top} = QRQ^{\top}.$$

In practice, due to the truncation of variables necessary to keep these within the accepted limits during the transform stages, the residual signal obtained by means of the proposed encoder scheme is different than the signal obtained when using conventional

schemes. It is worth clarifying how such truncations affect both schemes in Figure 5.1 (a) and (b). A 16-bit representation of variables between and after the transform stages is supported in HEVC. The binary shifts used in conventional HEVC for 8-bit input data representation were previously reported in Table 2-A, for the two stages of DCT transform. Note that the output dynamic ranges in each stage only depend on TU size and base matrix, but are uniform otherwise.

Assume that the same 16-bit representation is required when using the scheme in Figure 5.1 (b). The variables input to the transform stage are in this case unsigned 8-bit integers whose dynamic range is between 0 and 255, as opposite to residual samples which instead must also bring information on the sign. Assume now for instance that a $4 \times 4$ block is being transformed with DCT, using the corresponding base matrix:

$$Q_4 = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & -36 & -83 & 36 \\ 64 & -64 & 64 & -64 \\ 36 & 83 & -36 & -83 \end{bmatrix}.$$

The dynamic range that can be output by transforming the left-most column in the current block spans from 0 (obtained when the column contains only zeros) to +65280 (obtained when the column contains only 255s). On the contrary, the dynamic range that can be output by transforming the third column from the left is different and spans from $-32640$ (for an input column containing $[0, 255, 0, 255]^\top$) to +32640 (for an input column containing $[255, 0, 255, 0]^\top$). While the number of symbols that need to be represented is the same, the actual output dynamic range is different among different columns. To take advantage of the reduced dynamic range obtained as an effect of using the scheme in Figure 5.1 (b) which would require 1 less bit in the binary shifts than the values needed for the scheme in (a) reported in Table 2-A, such outputs should be appropriately offset and mapped to the same set of symbols. Such offsets would then influence the dynamic ranges in the next stage of transform. For this reason, the same adjustments used in conventional HEVC were used in this work also when considering the scheme in Figure 5.1 (b).

Some tests were performed to quantify the effects of using the modified encoder scheme compared with conventional HEVC. While the analysis and methods are mainly presented in the context of encoding of video sequences, they only directly affect intra-prediction and as such they can be also tested on still images. For this reason these initial tests were only performed on still images. In particular the Kodak test set [125] was used at this purpose, comprising 24 images of size $768 \times 512$ samples.

The compression performance was measured in terms of the BD-rate, where conventional HEVC was used as the anchor. Tests were performed under high quality constraints using QP values equal to 2, 5, 7 and 12 respectively. Note that using the schemes in Figure 5.1 (b) and (d) implies that the transform operation is performed twice for each block (both at encoder and decoder side), instead of only once as with the schemes in (a) and (c). This additional transform clearly adds some computational complexity to the encoding and decoding. The computational complexity of the proposed encoder and decoder were measured for each test image, and compared with the complexity of conventional HEVC encoder and decoder in terms of additional coding time, in percentage.

Results of these tests are reported in Table 5-A. In average, a negligible 0.1% BD-rate difference was obtained between the two codecs, with minimum and maximum BD-rates of respectively $-0.18\%$ and $+0.31\%$. In terms of computational complexity, in average a 6.7% increase in coding time was reported for both encoding time and decoding time. Using this scheme has clearly negligible effects on the coding performances and acceptable impacts in terms of complexity, while it provides the essential base for the analysis and proposed method presented in the rest of this chapter.

### 5.1.3 Per-coefficient Intra-prediction Dissimilarity

In general, by providing a more accurate prediction of the current block, a better encoder performance can be expected (due to the smaller residual samples which require less bits

Table 5-A: Comparison of proposed encoder and decoder schemes and conventional HEVC in terms of BD-rates, additional encoder time and additional decoder time.

| **Image** | **BD-rates (%)** | **Enc. Time (%)** | **Dec. Time (%)** |
|---|---|---|---|
| Stone building | 0.14 | 6.8 | 9.7 |
| Red door | −0.02 | 4.2 | 1.0 |
| Hats | 0.03 | 8.4 | 8.0 |
| Girl in red | 0.14 | 6.2 | 5.0 |
| Motocross bikes | 0.37 | 6.3 | 5.8 |
| Sailboat | 0.07 | 7.3 | 7.2 |
| Window | 0.16 | 7.2 | 2.1 |
| Market place | 0.44 | 5.1 | 7.1 |
| Spinnakers | −0.18 | 6.8 | 7.8 |
| Sailboat race | 0.10 | 6.3 | 5.7 |
| Pier | −0.17 | 2.8 | 8.3 |
| Couple on beach | 0.26 | 7.9 | 7.4 |
| Mountain stream | 0.14 | 6.8 | 13.5 |
| Water rafters | 0.13 | 4.8 | 3.8 |
| Girl | 0.20 | 0.9 | 3.0 |
| Tropical key | −0.05 | 6.4 | 6.4 |
| Monument | −0.10 | 6.3 | 3.8 |
| Model in black | 0.05 | 8.2 | 5.9 |
| Lighthouse | 0.31 | 11.1 | 10.8 |
| Mustang | 0.15 | 11.1 | 5.4 |
| Portland headlight | 0.11 | 7.9 | 5.1 |
| Barn and pond | 0.11 | 7.6 | 6.6 |
| Parrots | 0.07 | 7.5 | 15.6 |
| Chalet | 0.13 | 7.8 | 5.0 |

to be coded). While common distortion metrics in the spatial domain such as the SAD or SSD can be used to estimate the accuracy of a prediction, these types of metrics fail in measuring the impact of each intra-prediction method on different frequency components of the signals. It is instead reasonable to expect particular effects of certain prediction modes on specific frequency components. These effects can be captured and analysed to formulate appropriate processing methods to improve the coding efficiency.

In particular consider that the modified encoder scheme in Figure 5.1 (b) is used to encode a certain input signal. Denote as $K_{N,mode}$ the number of $N \times N$ prediction blocks considered by the encoder using the intra-prediction mode *mode* while coding this signal. Each of these blocks is transformed prior to the residual computation. Refer to such

transformed blocks as $\widetilde{P}_i$, where $i = 0, 1, ..., K-1$, and denote as $\widetilde{X}_i$ the corresponding transformed original blocks. Finally denote as $\widetilde{x}_i(m,n)$ and $\widetilde{p}_i(m,n)$ the samples at location $(m,n)$ in the blocks $\widetilde{X}_i$ or $\widetilde{P}_i$ respectively.

The problem becomes then that of studying the similarity (or conversely the dissimilarity) between the arrays $\widetilde{x}_i(m,n)$ and $\widetilde{p}_i(m,n)$, $i = 0, 1, ..., K-1$, for a given location $(m,n)$, intra-prediction mode *mode* and size $N$. Such problem is not trivial mainly due to two factors. First, due to the nature of the transformed signals, it is reasonable to assume that a large number of elements in the arrays $\widetilde{X}_i$ or $\widetilde{P}_i$ may be very close (if not identical) to zero. Common metrics such as the correlation coefficient or the Cosine distance are not well suited for this kind of data. None of these methods is defined for the case when one of the two signals is entirely zero valued. Second, the average values of the energy of the two signals vary greatly depending on the location $(m,n)$. Euclidean distance would not take into consideration such variations hence providing unreliable results when comparing similarity of arrays extracted at different locations. An attempt to normalise the distance by the energy or the mean of one of the two signals would produce similar results.

A tool capable to address both problems is represented by the Bray-Curtis coefficient (or Dice coefficient) [126], a well-known similarity measure widely used in image processing and pattern recognition applications. Bray-Curtis metrics were shown to perform best among several similarity metrics when used in texture image retrieval applications [127]. Also, as opposite to the Euclidean distance, they weight variables differently according to the energy of the signals hence avoiding problems due to the presence of sparse components especially in the transformed prediction signal. The coefficient is defined as twice the inner product between the signals, normalised by the sum of their energies:

$$c_{\{N,mode\}}(m,n) = \frac{\sum_{k=0}^{K-1} (\widetilde{p}_i(m,n)\widetilde{x}_i(m,n))}{\sum_{k=0}^{K-1} \widetilde{p}_i(m,n)^2 + \sum_{k=0}^{K-1} \widetilde{x}_i(m,n)^2}. \tag{5.1}$$

The Bray-Curtis dissimilarity is then defined as:

$$d_{\{N,mode\}}(m,n) = 1 - c_{\{N,mode\}}(m,n)\,. \tag{5.2}$$

The Bray-Curtis metrics are defined between 0 and 1. When using Equation 5.2, values close to 0 mean that the two signals are very similar one to the other, whereas higher values close to 1 mean that the prediction signal is almost completely different than the original at location $(m,n)$ in the transformed blocks.

Such metric was used to perform an analysis on the intra-prediction modes available in conventional HEVC. In particular, the modified encoder scheme in Figure 5.1 (b) was implemented in the context of HEVC intra-prediction. The same test images used in Table 5-A were coded using such a modified encoder. Coding was performed under very high quality constrains, namely the QP was set to 5. All pairs of transformed original and prediction blocks computed during the encoding were collected and grouped in terms of the transform size and intra-prediction mode used. Finally, the dissimilarity was computed as in Equation 5.1 and 5.2 for all locations $(m,n)$, for each possible block size $N$ and intra-prediction mode *mode*.

An extract of the results of this analysis is reported in this section in Table 5-B. For the sake of brevity, the Bray-Curtis dissimilarity is only presented for block sizes of $8 \times 8$ and $16 \times 16$, and for a limited number of modes in a selection of locations. Presenting the values at each single location and for all modes would be unnecessarily redundant and require inconveniently large tables. In particular only 19 locations are displayed in such tables, as shown in Figure 5.2 for the case of $16 \times 16$ blocks. A similar selection is used for $8 \times 8$ blocks: the block is subsampled to a grid of 16 values, and the three corner values are also considered in addition to this grid. As for the modes, only DC, planar, and modes equal or close to pure horizontal and pure vertical directions are shown. Dissimilarities smaller than 0.5 are highlighted in bold.

Finally, results are also presented for $4 \times 4$ blocks in Table 5-C. Only locations in the corners of the block are displayed in the table (namely locations (a), (e), (r) and (s) in Figure 5.2). Again, values of the dissimilarity smaller than 0.5 are highlighted in bold.

A first conclusion can be immediately highlighted from such tables: the DC coefficient (denoted as (a) in the tables) presents zero dissimilarity in all modes, for all transform sizes. This is expected in that all intra-prediction modes target efficient prediction of this coefficient and are tailored exactly for this purpose.

Clearly the size of the blocks has also an evident impact on the performance of intra-prediction. The fact that intra-prediction works better on smaller blocks is a well-known behaviour which can be easily explained by the fact that most intra-prediction techniques are based on the projection of reference samples in the prediction block. In smaller blocks these references are closer to the locations in which they are projected, therefore it can be expected that they are more correlated with the original content of such locations.

Only the sample in the bottom-right corner (namely the location at highest frequency,

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (a) | | | | (b) | | | | (c) | | | | (d) | | | (e) |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| (f) | | | | (g) | | | | (h) | | | | (i) | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| (j) | | | | (k) | | | | (l) | | | | (m) | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| (n) | | | | (o) | | | | (p) | | | | (q) | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| (r) | | | | | | | | | | | | | | | (s) |

Figure 5.2: Sample locations for displaying dissimilarity values.

denoted as (s) in the table) returned dissimilarity values higher than 0.5 in $4 \times 4$ blocks in some modes. This is opposite to the results obtained for larger block sizes, in which only a minority of locations provided dissimilarity values smaller than 0.5.

Table 5-B: Dissimilarity between transformed original samples and transformed prediction samples at selected locations in $8\times8$ and $16\times16$ blocks.

| ***mode*** | $(a)$ | $(b)$ | $(c)$ | $(d)$ | $(e)$ | $(f)$ | $(g)$ | $(h)$ | $(i)$ | $(j)$ | $(k)$ | $(l)$ | $(m)$ | $(n)$ | $(o)$ | $(p)$ | $(q)$ | $(r)$ | $(s)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $8\times8$ **blocks** | | | | | | | | | | | | | | | | | | |
| Planar | **0.00** | 0.75 | 0.85 | 0.96 | 0.98 | 0.74 | 1.00 | 1.00 | 1.00 | 0.86 | 1.00 | 1.00 | 1.00 | 0.96 | 1.00 | 1.00 | 1.00 | 0.98 | 1.00 |
| DC | **0.00** | 0.92 | 0.93 | 0.95 | 0.96 | 0.92 | 0.91 | 0.92 | 0.94 | 0.95 | 0.92 | 0.93 | 0.94 | 0.94 | 0.93 | 0.95 | 0.96 | 0.95 | 0.98 |
| 7 | **0.00** | 0.99 | 1.00 | 1.00 | 1.00 | **0.35** | 0.92 | 0.98 | 0.99 | 0.89 | 0.80 | 0.94 | 0.96 | 0.98 | 0.73 | 0.98 | 0.99 | 0.99 | 0.96 |
| 8 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.26** | 0.99 | 1.00 | 1.00 | 0.59 | 0.95 | 0.99 | 1.00 | 0.99 | 0.86 | 0.94 | 0.99 | 0.88 | 0.99 |
| 9 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.22** | 1.00 | 1.00 | 1.00 | **0.36** | 1.00 | 1.00 | 1.00 | 0.54 | 1.00 | 1.00 | 1.00 | 0.58 | 1.00 |
| Pure Hor. | **0.00** | 0.93 | 0.93 | 0.94 | 0.94 | **0.18** | 0.88 | 0.91 | 0.93 | **0.25** | 0.91 | 0.92 | 0.93 | **0.32** | 0.93 | 0.95 | 0.96 | **0.36** | 0.98 |
| 11 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.23** | 1.00 | 1.00 | 1.00 | **0.38** | 1.00 | 1.00 | 1.00 | **0.46** | 1.00 | 1.00 | 1.00 | 0.53 | 1.00 |
| 12 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.29** | 0.99 | 1.00 | 1.00 | 0.57 | 0.95 | 0.99 | 1.00 | 0.84 | 0.88 | 0.95 | 0.99 | 0.91 | 0.99 |
| 13 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.36** | 0.91 | 0.98 | 0.98 | 0.80 | 0.79 | 0.94 | 0.93 | 0.98 | 0.72 | 0.98 | 1.00 | 0.97 | 0.97 |
| 23 | **0.00** | **0.35** | 0.83 | 0.98 | 0.97 | 1.00 | 0.93 | 0.89 | 0.84 | 1.00 | 0.99 | 0.95 | 1.00 | 1.00 | 0.99 | 0.97 | 0.97 | 1.00 | 0.98 |
| 24 | **0.00** | **0.26** | 0.59 | 0.95 | 0.83 | 1.00 | 0.99 | 0.97 | 0.92 | 1.00 | 1.00 | 0.99 | 0.97 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 | 0.99 |
| 25 | **0.00** | **0.19** | **0.33** | **0.44** | **0.49** | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Pure ver. | **0.00** | **0.15** | **0.23** | **0.29** | **0.29** | 0.92 | 0.87 | 0.88 | 0.92 | 0.93 | 0.88 | 0.92 | 0.93 | 0.94 | 0.90 | 0.93 | 0.95 | 0.94 | 0.98 |
| 27 | **0.00** | **0.18** | **0.30** | **0.43** | **0.46** | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 28 | **0.00** | **0.24** | 0.57 | 0.95 | 0.87 | 1.00 | 0.98 | 0.94 | 0.90 | 1.00 | 1.00 | 0.98 | 0.95 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 | 0.99 |
| 29 | **0.00** | **0.33** | 0.80 | 1.00 | 1.00 | 0.99 | 0.90 | 0.82 | 0.79 | 1.00 | 0.98 | 0.94 | 0.99 | 1.00 | 0.98 | 0.97 | 0.99 | 1.00 | 0.97 |
| | $16\times16$ **blocks** | | | | | | | | | | | | | | | | | | |
| Planar | **0.00** | 0.77 | 0.86 | 0.96 | 0.99 | 0.75 | 1.00 | 1.00 | 1.00 | 0.87 | 1.00 | 1.00 | 1.00 | 0.96 | 1.00 | 1.00 | 1.00 | 0.98 | 0.99 |
| DC | **0.00** | 0.97 | 0.96 | 0.98 | 0.99 | 0.95 | 0.96 | 0.96 | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.96 | 0.97 | 0.98 | 0.98 | 0.97 | 1.00 |
| 6 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | 0.91 | 0.98 | 1.00 | 0.97 | 0.98 | 0.86 | 1.00 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.96 | 1.00 |
| 7 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | 0.69 | 0.99 | 0.98 | 1.00 | 0.96 | 0.99 | 1.00 | 0.99 | 0.98 | 0.97 | 0.99 | 1.00 | 0.96 | 1.00 |
| 8 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.44** | 0.99 | 1.00 | 1.00 | 0.92 | 0.95 | 0.99 | 0.99 | 0.99 | 0.99 | 1.00 | 1.00 | 0.97 | 1.00 |
| 9 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.26** | 1.00 | 1.00 | 1.00 | **0.46** | 1.00 | 1.00 | 1.00 | 0.95 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 |
| Pure Hor. | **0.00** | 0.96 | 0.96 | 0.97 | 0.98 | **0.18** | 0.95 | 0.97 | 0.97 | **0.21** | 0.96 | 0.95 | 0.97 | **0.26** | 0.96 | 0.97 | 0.98 | **0.42** | 1.00 |
| 11 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.29** | 1.00 | 1.00 | 1.00 | 0.50 | 1.00 | 1.00 | 1.00 | 0.62 | 1.00 | 1.00 | 1.00 | 0.98 | 1.00 |
| 12 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | **0.42** | 0.98 | 1.00 | 1.00 | 0.89 | 0.97 | 1.00 | 0.99 | 0.96 | 0.98 | 0.99 | 1.00 | 0.97 | 1.00 |
| 13 | **0.00** | 1.00 | 1.00 | 1.00 | 1.00 | 0.56 | 0.95 | 0.98 | 0.99 | 0.94 | 0.99 | 1.00 | 0.98 | 0.96 | 0.94 | 0.99 | 1.00 | 0.97 | 1.00 |
| 23 | **0.00** | 0.61 | 0.92 | 0.95 | 0.97 | 0.99 | 0.95 | 0.99 | 0.95 | 1.00 | 0.97 | 1.00 | 0.99 | 1.00 | 1.00 | 0.99 | 1.00 | 0.99 | 1.00 |
| 24 | **0.00** | **0.44** | 0.90 | 0.96 | 0.97 | 0.99 | 0.98 | 0.95 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 0.99 |
| 25 | **0.00** | **0.24** | **0.45** | 0.73 | 0.89 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 0.99 |
| Pure Ver. | **0.00** | **0.15** | **0.25** | **0.29** | **0.39** | 0.96 | 0.95 | 0.96 | 0.97 | 0.97 | 0.96 | 0.96 | 0.96 | 0.97 | 0.96 | 0.97 | 0.98 | 0.97 | 1.00 |
| 27 | **0.00** | **0.20** | **0.38** | 0.75 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 |
| 28 | **0.00** | **0.38** | 0.85 | 1.00 | 0.99 | 1.00 | 0.98 | 0.95 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 | 1.00 | 1.00 |
| 29 | **0.00** | 0.58 | 0.93 | 0.99 | 0.94 | 1.00 | 0.95 | 1.00 | 0.97 | 1.00 | 0.99 | 0.99 | 0.99 | 1.00 | 1.00 | 0.99 | 1.00 | 1.00 | 0.99 |

More interestingly though, the dissimilarity shows a strong correlation with the intra-prediction mode being used. Angular modes close to the pure vertical direction ($mode = 26$ following HEVC nomenclature) all show relatively low values of the dissimilarity in locations close to the left-most column in the blocks. Conversely, modes close to the pure horizontal direction ($mode = 10$) show relatively low dissimilarity values close to the top-most row in the blocks.

The results in Table 5-B show that conventional intra-prediction modes may provide very poor prediction accuracy of high frequency components of the signals. Especially in large blocks, only a restricted number of coefficients at low frequencies are efficiently predicted. Conventional video coding architectures do not take into consideration such

Table 5-C: Dissimilarity between transformed original samples and transformed prediction samples at selected locations in $4 \times 4$ blocks.

| ***mode*** | $(a)$ | $(e)$ | $(r)$ | $(s)$ |
|---|---|---|---|---|
| Planar | **0.00** | **0.02** | **0.04** | **0.49** |
| DC | **0.00** | **0.02** | **0.04** | **0.48** |
| 2 | **0.00** | **0.02** | **0.04** | 0.92 |
| 3 | **0.00** | **0.02** | **0.03** | 0.65 |
| 4 | **0.00** | **0.02** | **0.04** | 0.53 |
| 5 | **0.01** | **0.03** | **0.05** | **0.50** |
| 6 | **0.00** | **0.02** | **0.05** | 0.52 |
| 7 | **0.00** | **0.02** | **0.08** | 0.51 |
| 8 | **0.00** | **0.02** | **0.08** | **0.46** |
| 9 | **0.00** | **0.02** | **0.08** | **0.45** |
| Pure Hor. | **0.00** | **0.02** | **0.05** | **0.30** |
| 11 | **0.00** | **0.02** | **0.07** | **0.47** |
| 12 | **0.00** | **0.02** | **0.08** | **0.48** |
| 13 | **0.00** | **0.02** | **0.08** | 0.54 |
| 14 | **0.00** | **0.02** | **0.06** | 0.53 |
| 15 | **0.00** | **0.02** | **0.04** | 0.52 |
| 16 | **0.00** | **0.02** | **0.03** | 0.72 |
| 17 | **0.00** | **0.02** | **0.03** | 0.62 |
| 18 | **0.00** | **0.02** | **0.03** | 0.92 |
| 19 | **0.00** | **0.03** | **0.03** | 0.65 |
| 20 | **0.00** | **0.03** | **0.02** | 0.73 |
| 21 | **0.00** | **0.03** | **0.03** | 0.50 |
| 22 | **0.00** | **0.05** | **0.03** | 0.57 |
| 23 | **0.00** | **0.06** | **0.03** | 0.54 |
| 24 | **0.00** | **0.07** | **0.03** | 0.51 |
| 25 | **0.00** | **0.06** | **0.03** | **0.48** |
| Pure Ver. | **0.00** | **0.03** | **0.02** | **0.30** |
| 27 | **0.00** | **0.07** | **0.03** | **0.47** |
| 28 | **0.00** | **0.07** | **0.02** | **0.48** |
| 29 | **0.00** | **0.07** | **0.02** | 0.55 |
| 30 | **0.00** | **0.05** | **0.02** | 0.51 |
| 31 | **0.00** | **0.03** | **0.02** | **0.46** |
| 32 | **0.00** | **0.03** | **0.03** | 0.54 |
| 33 | **0.00** | **0.02** | **0.03** | 0.62 |
| 34 | **0.00** | **0.03** | **0.04** | 0.85 |

a behaviour in that they consider with equal importance all coefficients at all frequency components, following from the assumption that poorly predicted coefficients at high frequencies can be discarded during quantisation. This cannot be allowed under high quality constraints. By computing independently the transform of the prediction and original blocks, novel approaches can be developed to improve compression efficiency under these constraints, as illustrated in the rest of this chapter.

## 5.2 Transform Domain Prediction Processing

### 5.2.1 Masking patterns

The analysis in the previous section is helpful in determining which frequency components of the prediction signal should be preserved, and which others may instead be discarded as they are not representative of the original signal. On the other hand such an analysis gives no information regarding the real content of the original blocks at different frequency components.

In order to study the behaviour of these frequency components, the coefficient values of the original signal can be studied taking into consideration which intra-prediction mode would be selected by a conventional HEVC encoder. As an example the plots in Figure 5.4 show histograms reporting the frequency of occurrence of coefficient values in the original signal for some locations in the transformed block, for the case of $8 \times 8$ blocks that would be intra-predicted with horizontal mode 9. Locations are identified using same nomenclature as previously defined in Figure 5.2.

It is reasonable to expect that the content in blocks that are well predicted by the almost horizontal mode 9 presents a strong directionality. Such directionality reflects in larger coefficients toward the left-most portion in the blocks, and conversely smaller coefficients in the right-most portion in the blocks, as evident from the histograms in Figure 5.4. More than 50% of coefficients are valued between $-25$ and 25 in locations

$(d)$, $(i)$, $(m)$ and $(q)$. Interestingly such frequency components are also characterised by very high values of the dissimilarity as reported in Table 5-B. This means that while the original signal is likely to assume very small values at these frequency components, the corresponding coefficients in the prediction signal bring almost no resemblance with such values.

In order to take into consideration this behaviour, the encoder and decoder schemes can be further modified to include an additional step of processing which acts on a selection of the frequency components in the transformed prediction blocks before the residual computation. The resulting modified encoder and decoder schemes are illustrated in Figure 5.3.

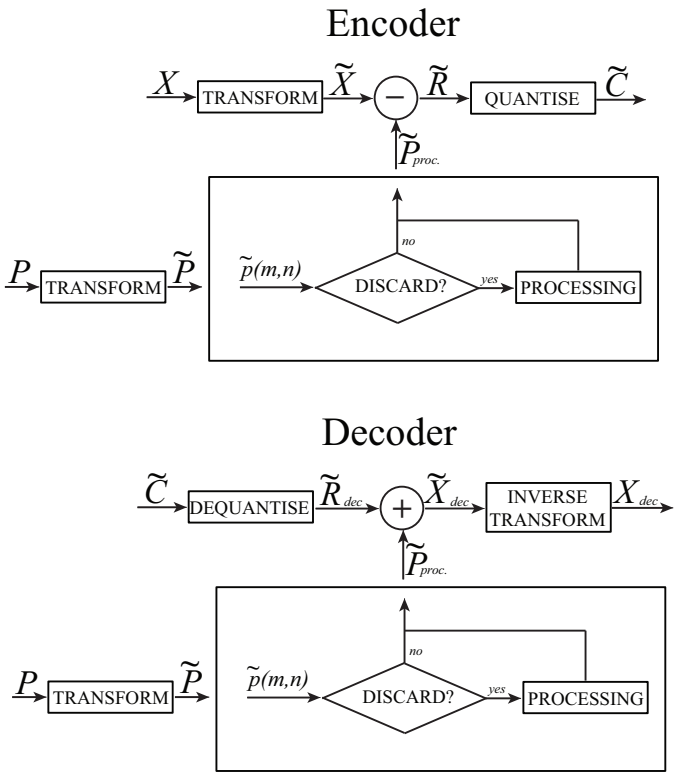



Figure 5.3: Proposed encoder and decoder schemes including processing of the transformed prediction blocks.

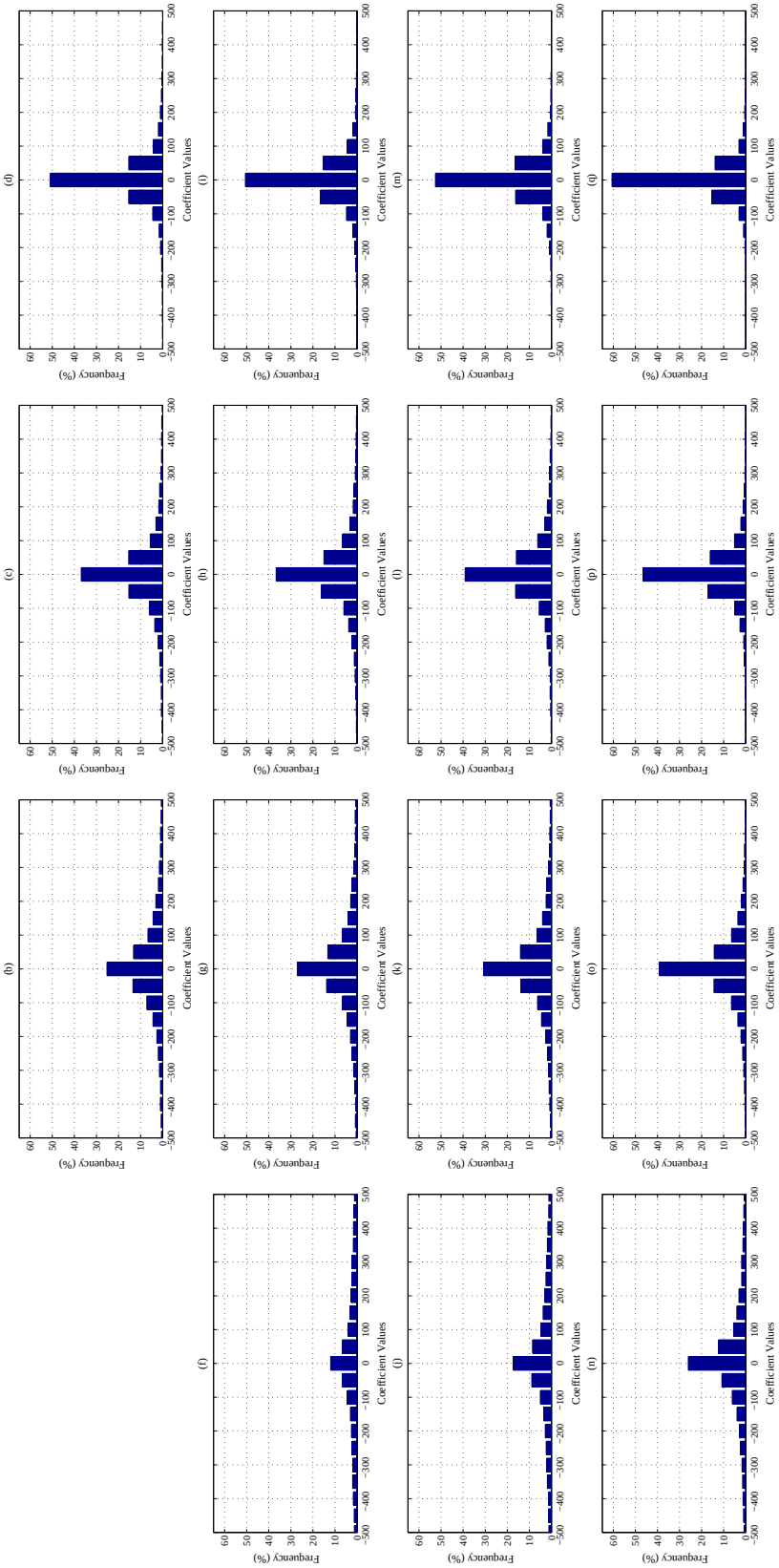



Figure 5.4: Frequency of occurrence of coefficient values at different locations in the transformed blocks, for intra-prediction mode 9, block size $8 \times 8$.

The process of selecting particular coefficients to discard in the transformed prediction block can be easily formalised through the definition of a set of masking matrices, referred to as patterns in the rest of this chapter. A pattern is a matrix $H$ of a given size $N \times N$, whose elements $h(m, n)$ are binary variables (namely either 1 or 0). The value of an element in a certain location determines whether the corresponding coefficient in the transformed prediction block is preserved or it is discarded and processed as in Figure 5.3. For instance, it is clear from results in Table 5-B that the block may be split vertically into two portions, where only coefficients in the right-most portion of the prediction block are preserved.

Four classes of patterns are defined. Formally consider an integer parameter $L$, referred to as the pattern size, where $0 \leq L \leq N$. Three values of $L$ are considered, namely $L = N/4$, $L = N/2$ and $L = 3N/4$. Then the following patterns are considered:

1. Vertical rectangular patterns, referred to as $H_{vr}$, consist of $L$ consecutive rows of preserved coefficients in the top-side portion of the block, or:

$$h_{(vr,L)}(m, n) = \begin{cases} 1, & \text{if } n \leq L; \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

2. Horizontal rectangular patterns, referred to as $H_{hr}$, consist of $L$ consecutive columns of preserved coefficients in the left-side portion of the block ,or:

$$h_{(hr,L)}(m, n) = \begin{cases} 1, & \text{if } m \leq L; \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

3. Square patterns, referred to as $H_{sq}$, consist of $L \times L$ preserved coefficients in the top-left portion of the block, or:

$$h_{(sq,L)}(m, n) = \begin{cases} 1, & \text{if } m \leq L \text{ and } n \leq L; \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

4. Triangular patterns, referred to as $H_{tr}$, consist of a triangular region of preserved coefficients in the top-left portion of the pattern, or:

$$h_{(tr,L)}(m,n) = \begin{cases} 1, & \text{if } (m+n) \leq N; \\ 0, & \text{otherwise.} \end{cases} \tag{5.6}$$

A certain pattern $H$ is applied to a transformed prediction block $\widetilde{P}$ by Hadamard (entry-wise) product. Some example patterns are shown in Figure 5.5.

The schemes in Figure 5.3 were implemented again in the context of HEVC, although in theory they can be adapted to any video codec making use of the block-based hybrid model. In order to identify which patterns should be used depending on characteristics of the current block and prediction block, first experiments were performed under the assumption that the processing in the schemes in the figure is performed by simply setting the discarded masked coefficients to zero, or $\widetilde{p}(m,n) = 0$ if $h(m,n) = 0$.

The following algorithm, referred to as Algorithm 1, was then implemented at the encoder side. A list of all considered patters $H_1$,$H_2$, ...,$H_M$ is considered, where $M$ is the number of available patterns; an additional element $H_0$ was included at the first position in the list to identify the trivial pattern, where $h_0(m,n) = 1$ for $m = 0, ...N-1$, $n = 0, ...N-1$, namely this is the case when no coefficients are discarded in the prediction block. After a block of samples is intra-predicted using a given mode, prediction and original signals are independently transformed obtaining $\widetilde{P}$ and $\widetilde{X}$ respectively. An index

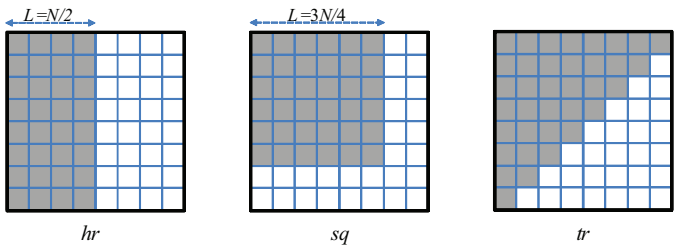



Figure 5.5: Example of patterns used for transform domain prediction processing. Coefficients in shaded locations are preserved, coefficients in white locations are discarded.

$j$ is initialised to zero and:

1. The pattern $H_j$ is extracted from the list and applied to $\widetilde{P}$ to obtain $\widetilde{P}_{proc}$. The residual samples are computed as $\widetilde{R} = \widetilde{X} - \widetilde{P}_{proc}$ and quantised to obtain $\widetilde{C}$. This is dequantised and inverse transformed to obtain $\widetilde{R}_{dec}$, which is finally used to compute the reconstruction $\widetilde{X}_{rec} = \widetilde{R}_{dec} + \widetilde{P}_{proc}$.

2. $\widetilde{C}$ and $\widetilde{X}_{rec}$ are used to compute the RD cost relative to the current element $j$. A temporary solution is considered as the index $j^o$ such that pattern $H_{j^o}$ corresponds to minimum RD cost.

3. If $j < M$, the index $j$ is incremented and the algorithm goes back to step 1. Otherwise the pattern at minimum RD cost is output, identified by its optimal index $j^o$.

The index $j^o$ to select the correct pattern in the list is signalled to the decoder in the bitstream for each block in which the algorithm is enabled. At the decoder side, such index is decoded and used to extract $H_{j^o}$. This is then applied to the transformed prediction as in the scheme at the bottom of Figure 5.3.

The approach was tested again in the Kodak image test set. In Table 5-D, the first and second most selected classes of patterns (regardless of the size $L$) are presented together with their frequency of occurrence, for a selection of available HEVC intra-prediction modes. Clearly the patterns are chosen according to the directionality of the content in the blocks. Horizontal patterns are most likely selected in horizontal modes, and vertical patterns are most likely selected in vertical modes. The trivial pattern (namely when no coefficients are discarded) is the most chosen pattern in the majority of the cases, and it frequency of occurrence increases as the size of the blocks decrease. In $4 \times 4$ TUs it is chosen in average in 50% of the cases.

### 5.2.2 Replacing Coefficients with Context-aware Look-up Tables

Similar histograms as in Figure 5.4 can be generated for any intra-prediction mode and any block size. From these data it is clear that while a large number of coefficients in the original block are close to zero, yet a significant number of blocks present non zero valued coefficients even at high frequencies. For instance in the case illustrated in Figure 5.4 (corresponding to the almost horizontal mode 9 and block size $8 \times 8$), more than 30% of the highest frequency coefficients (at location $(q)$ in the figure) resulted in an absolute value between 25 and 75.

These large coefficients at high frequencies are common in areas of the currently encoded picture which contain large amount of details or that are finely textured. Such fine details result in a great deal of high frequency content, which is difficult to predict using information previously encoded in the signal. It is unlikely that replicas of the signal can be found with such a level of granularity, which means that conventional techniques are likely to fail in providing a good prediction of these components. Moreover,

Table 5-D: Most selected patterns for each intra-prediction mode.

| | $4 \times 4$ | | | | $8 \times 8$ | | | |
|---|---|---|---|---|---|---|---|---|
| ***mode*** | **pattern** | **freq. (%)** | **pattern** | **freq. (%)** | **pattern** | **freq. (%)** | **pattern** | **freq. (%)** |
| Planar | $H_0$ | 55.47 | $H_{tr}$ | 24.29 | $H_0$ | 24.47 | $H_{vr}$ | 23.52 |
| DC | $H_0$ | 53.80 | $H_{tr}$ | 24.69 | $H_{vr}$ | 24.88 | $H_{hr}$ | 23.18 |
| 7 | $H_0$ | 50.46 | $H_{tr}$ | 24.88 | $H_{vr}$ | 24.87 | $H_0$ | 24.19 |
| 8 | $H_0$ | 49.32 | $H_{tr}$ | 25.46 | $H_{vr}$ | 24.92 | $H_{sq}$ | 23.51 |
| 9 | $H_0$ | 51.10 | $H_{tr}$ | 24.68 | $H_0$ | 29.57 | $H_{hr}$ | 22.82 |
| Pure Hor. | $H_0$ | 56.08 | $H_{tr}$ | 22.84 | $H_{hr}$ | 27.17 | $H_0$ | 23.18 |
| 11 | $H_0$ | 50.98 | $H_{tr}$ | 24.79 | $H_0$ | 28.23 | $H_{hr}$ | 23.72 |
| 12 | $H_0$ | 50.76 | $H_{tr}$ | 25.16 | $H_{vr}$ | 24.76 | $H_{sq}$ | 23.37 |
| 13 | $H_0$ | 50.18 | $H_{tr}$ | 25.41 | $H_{vr}$ | 24.48 | $H_{sq}$ | 23.44 |
| 23 | $H_0$ | 50.78 | $H_{tr}$ | 25.37 | $H_0$ | 26.17 | $H_{sq}$ | 22.63 |
| 24 | $H_0$ | 51.69 | $H_{tr}$ | 25.56 | $H_{sq}$ | 23.88 | $H_{hr}$ | 23.54 |
| 25 | $H_0$ | 52.11 | $H_{tr}$ | 25.43 | $H_0$ | 28.03 | $H_{vr}$ | 25.49 |
| Pure Ver. | $H_0$ | 56.95 | $H_{tr}$ | 23.50 | $H_{vr}$ | 30.26 | $H_0$ | 22.31 |
| 27 | $H_0$ | 50.83 | $H_{tr}$ | 25.82 | $H_0$ | 27.99 | $H_{vr}$ | 26.44 |
| 28 | $H_0$ | 51.81 | $H_{tr}$ | 25.65 | $H_0$ | 23.93 | $H_{hr}$ | 23.35 |
| 29 | $H_0$ | 52.22 | $H_{tr}$ | 25.17 | $H_0$ | 25.44 | $H_{hr}$ | 23.40 |
| | $16 \times 16$ | | | | $32 \times 32$ | | | |
| ***mode*** | **pattern** | **freq. (%)** | **pattern** | **freq. (%)** | **pattern** | **freq. (%)** | **pattern** | **freq. (%)** |
| Planar | $H_{vr}$ | 26.00 | $H_{hr}$ | 24.65 | $H_{vr}$ | 27.18 | $H_{hr}$ | 26.16 |
| DC | $H_{hr}$ | 26.16 | $H_{vr}$ | 25.81 | $H_{vr}$ | 31.21 | $H_{sq}$ | 23.00 |
| 7 | $H_{sq}$ | 24.65 | $H_{hr}$ | 24.07 | $H_{vr}$ | 27.50 | $H_0$ | 24.50 |
| 8 | $H_{hr}$ | 25.03 | $H_{vr}$ | 24.99 | $H_{vr}$ | 29.68 | $H_0$ | 23.19 |
| 9 | $H_{hr}$ | 30.96 | $H_{sq}$ | 26.43 | $H_{vr}$ | 25.63 | $H_{hr}$ | 24.43 |
| Pure Hor. | $H_{hr}$ | 27.78 | $H_{sq}$ | 22.68 | $H_{sq}$ | 35.99 | $H_{hr}$ | 27.13 |
| 11 | $H_{hr}$ | 29.60 | $H_{sq}$ | 26.92 | $H_{hr}$ | 27.41 | $H_{vr}$ | 23.95 |
| 12 | $H_{vr}$ | 25.60 | $H_{hr}$ | 23.18 | $H_{vr}$ | 28.93 | $H_{hr}$ | 24.44 |
| 13 | $H_{vr}$ | 24.88 | $H_{sq}$ | 24.69 | $H_{vr}$ | 29.65 | $H_{hr}$ | 23.12 |
| 23 | $H_{sq}$ | 28.07 | $H_{hr}$ | 23.52 | $H_0$ | 23.81 | $H_{vr}$ | 22.22 |
| 24 | $H_{vr}$ | 25.83 | $H_{hr}$ | 24.80 | $H_{hr}$ | 28.67 | $H_{vr}$ | 24.48 |
| 25 | $H_{sq}$ | 29.21 | $H_{vr}$ | 26.76 | $H_{vr}$ | 27.05 | $H_{sq}$ | 25.45 |
| Pure Ver. | $H_{vr}$ | 30.53 | $H_{sq}$ | 25.10 | $H_{sq}$ | 47.17 | $H_{vr}$ | 33.74 |
| 27 | $H_{vr}$ | 28.61 | $H_{sq}$ | 27.27 | $H_{vr}$ | 25.61 | $H_{hr}$ | 23.87 |
| 28 | $H_{vr}$ | 25.54 | $H_{hr}$ | 24.94 | $H_0$ | 24.58 | $H_{sq}$ | 24.17 |
| 29 | $H_{hr}$ | 27.92 | $H_{vr}$ | 22.35 | $H_{vr}$ | 26.99 | $H_{hr}$ | 22.70 |

also setting to zero these prediction coefficients is clearly not optimal.

If previously encoded content cannot provide a prediction of such non zero components, an alternative way of predicting these values consists in introducing new content in the prediction signal specifically with the intent of reducing the residuals at high frequencies. Such synthetic content can be considered as an external source of information with the goal of introducing additional spatial redundancy in the signal.

Synthetic content can be introduced in the signal studying histograms similar to those in Figure 5.4, which can be generated and studied for each intra-prediction mode and block size. The values in the histograms which appear with the highest relative frequency can be used as possible replacements for the discarded coefficients after the application of a given pattern. The problem is then that of formalising (and optimising) the process of replacing these coefficients with previously defined synthetic values. In order to do so, the concept of context-aware look-up tables was used in this chapter.

Context awareness is a fundamental component of video coding especially with regards to the entropy coding module. Typical methods such as CAVLC or CABAC are based on the definition of context-aware parameters (usually referred to as context models), which take into account the frequency of occurrence of symbols in the previously encoded signal and also local conditions in the signal, to optimise the encoding of the current symbol. In addition to the intrinsic context awareness of entropy coding methods, a further step of contextuality is also commonly used, usually for the purpose of encoding parameters which can assume values from a fixed dictionary, under the assumption that the frequency of occurrence of these values varies considerably depending on some predictable, local characteristics of the signal. To fully take advantage of context-aware entropy coding, the same parameter would require the definition of different context models depending on these local variations. This might lead to over modelling consequently compromising the coding performance.

In order to avoid such issues, the impact of local conditions on the outcomes of a

parameter can be embedded into appropriately designed context-aware look-up tables. In particular, a number of different look-up tables is defined to model the outcomes of the same parameter. Each look-up table is associated with a pre-determined set of conditions, which are known to have a predictable impact on the outcomes of the parameter. The look-up tables contain a sorted sequence of all symbols in the dictionary, identified by their order of appearance (the index). Symbols are sorted in such a way that the same index is associated in each look-up table with symbols which appear with approximately the same frequency, upon the occurrence of each pre-determined set of conditions. The encoder considers the information required to select the correct look-up table for a given parameter, and then extracts the corresponding index. Such index is encoded instead of the original symbol.

Context-aware look-up tables can be used in the methodology presented in this chapter for the purpose of replacing coefficients that are discarded when applying a certain pattern. A dictionary can be defined by considering a set of $T$ different values $\alpha_0, ..., \alpha_{T-1}$. These values are selected to be representative of the range spanned by the actual coefficients at high frequencies. A trade-off between frequency of occurrence of coefficients and their effects on the coding efficiency should be considered. While large coefficients tend to appear less often, they also have a higher impact on the related bitrates when they are not accurately predicted: in these cases very high residual samples are obtained, which are inefficiently compressed by conventional methods. For this reason it makes sense to include in the dictionary many small values, but also some sparse large values to deal with the cases when they might be needed. A total of 33 elements in the dictionary was considered in the implementation used in this thesis, where $\alpha_0 = 0$, and values span from $-128$ to $+128$.

In order to correctly select and use a certain element from the dictionary, this must be signalled in the bitstream so that the same element can be extracted and applied also at the decoder side as in the scheme at the bottom of Figure 5.3. Unfortunately, high bitrates may result as a consequence of this signalling especially if the number $T$ of

elements in the dictionary is large. For this reason, following again from the assumption that the frequency of occurrence of coefficient values is dependent on encoder decisions on the currently encoded block, it makes sense to restrict and adapt the number of allowed elements in the dictionary to these features. Instead of considering all possible dictionary elements in each block, subsets of such dictionary can be used in the form of context-aware look-up tables.

For each TU, a context $\Omega = \{N, H, mode\}$ is considered, where $N$ is the TU height and width, $H$ is the currently used pattern and *mode* is the intra-prediction mode being used. For each instance of $\Omega$, a look-up table $F_\Omega$ is defined as an indexed array $f_\Omega(k)$ where $k = 0, ..., K-1$; the $K$ elements in each table are a particular sub-set of the elements $\alpha_0, ..., \alpha_{T-1}$ in the dictionary. The length of the look-up tables $K$ can be set to allow the testing of a sufficient number of coefficient values in each block, while at the same time limiting the rates needed for transmitting the corresponding index. A value $K = 8$ was used in the implementation described in this chapter.

The $K$ elements to form each look-up table and their order can be derived from statistical analysis. In fact such elements should represent the entire range of values assumed by the coefficients in the transformed original blocks, spanning from very probable small values to more rare large values. In order to derive these statistics, experiments were performed on some test sequences as follows. For each transformed original block $\widetilde{X}$ with its corresponding context $\Omega\{N, H, mode\}$, all $T$ values $\alpha_0, ..., \alpha_{T-1}$ in the dictionary were tested. For a given $\alpha_i$ the block $\widetilde{P}_{proc,\alpha_i}$ was computed by applying pattern $H$ to the transformed prediction block $\widetilde{P}$ and then replacing all discarded coefficients with $\alpha_i$, or $\widetilde{p}_{proc,\alpha_i}(m,n) = \alpha_i$ if $h(m,n) = 0$. The element in the dictionary at minimum prediction distortion (computed using SAD) was selected as in:

$$\underset{\alpha_i}{\operatorname{argmin}} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} |\widetilde{p}_{proc,\alpha_i}(m,n) - \widetilde{x}(m,n)|$$

Given a feature set $\Omega = \{N, H, mode\}$, the probability of occurrence of each element

in the dictionary $P(\alpha_i|\Omega)$ was then estimated as the number of times the element $\alpha_i$ was selected over the total number of blocks coded with context $\Omega$.

A set of $K$ target probabilities $P_0$, $P_1$, ..., $P_{K-1}$ was also defined. Probability values spanning from 0.4 to 0.02 were used in the implementation described in this thesis. Finally for each instance of $\Omega$, $K$ elements were selected from the dictionary to be included in the look-up table $F_\Omega$: for each target probability $P_k$, $k = 0, ..., K-1$, the element $\alpha_i$ such that $P(\alpha_i|\Omega)$ is closer to $P_k$ was selected. To improve the efficiency of entropy coding when signalling the index to select the correct element in the look-up tables, elements are sorted by decreasing probability, or $P(f_\Omega(i)|\Omega) \geq P(f_\Omega(j)|\Omega)$ if $i < j$.

When coding a certain block, the context $\Omega$ is considered and the appropriate look-up table is used. Each element $f_\Omega(k)$ in the look-up table is tested in an RD sense, and finally the optimal index $k^o$ to identify the correct element is selected and transmitted in the bitstream to be used at the decoder side. Note that clearly all look-up tables must be available to both the encoder and decoder.

The two steps of the proposed approach described respectively in this subsection and in Subsection 5.2.1 can be eventually integrated within a single algorithm. The first step consists in classifying coefficients within the transformed prediction block by means of masking patterns, to select those that should be preserved and those that can be discarded. An appropriate pattern $H_{j^o}$ must be selected at this purpose for each block. In the second step, discarded coefficients are replaced with more meaningful synthetic content by means of context-aware look-up tables. The context $\Omega = \{N, H, mode\}$ is derived for the current block, and a look-up table $F_\Omega$ is considered. An element $f_\Omega(k^o)$ must be appropriately extracted, and finally $\widetilde{p}_{proc}(m,n) = f_\Omega(k^o)$ if $h_{j^o}(m,n) = 0$.

In theory these steps should be performed in such a way that the optimal combination of best pattern and best element in the corresponding look-up table is selected. Algorithm 1 as presented in Subsection 5.2.1 would need to be modified accordingly: a nested loop

to test the $K$ elements in $F_\Omega$ should be considered within the main loop described in Step 1 of such algorithm. Selecting and transmitting the optimal value of both indices $k$ and $j$ in an RD sense is not feasible, because it would likely result in very high bitrates, and it is also considerably expensive in terms of computational complexity. In total $(M \times K) + 1$ iterations would need to be performed (this also includes testing of the trivial pattern). Performing this number of iterations is not optimal even if very few elements are considered in the look-up tables.

To solve these issues a different approach can be formulated. Instead of selecting in an RD sense and transmitting in the bitstream the index $j^o$ to identify the best pattern, the choice of pattern can be fixed by means of statistical analysis, depending on features of the current block such as intra-prediction mode *mode* and TU size $N$. While more complex statistics might be used, for simplicity in this paper only intra-prediction mode and TU size were considered at this purpose. Table 5-D already presents the most frequently selected pattern for each combination of these features. Such pattern $H_{N,mode}$ as reported in the table can be used any time a block of size $N \times N$ is encoded using intra-prediction mode *mode*; in case the most frequently selected pattern for a given mode *mode* and TU size $N$ is the trivial pattern $H_0$, the second most frequently selected pattern is used instead.

Following from this restriction, the context reduces to $\Omega = \{N, mode\}$ due to the fact that the pattern depends on the other two features. Consequently a much smaller number of look-up tables need to be computed and stored among the encoder and decoder resources. Also, following this adaptation an algorithm to perform frequency domain prediction processing can be defined which only needs $K + 1$ iterations for each block, referred to as Algorithm 2 in the rest of this paper and defined as follows. After a block of samples $X$ is intra-predicted using a given mode, prediction and original signals are independently transformed obtaining $\widetilde{P}$ and $\widetilde{X}$ respectively. The pattern $H_{N,mode}$ is considered from Table 5-D. Also the look-up table $F_\Omega$ is extracted according to the block context. An index $k$ is initialised to zero. This is used to identify the elements

in the look-up tables, with the exception of a value $k = 0$, reserved to signal the case when the trivial pattern $H_0$ is used and no coefficient is discarded (and consequently no processing is performed).

Then:

1. If $k \neq 0$, the element $f_\Omega(k)$ is extracted from the look-up table. Pattern $H_{N,mode}$ is applied to $\widetilde{P}$ and $\widetilde{P}_{proc}$ is obtained as:

$$\widetilde{p}_{proc}(m,n) = \begin{cases} \widetilde{p}(m,n) & \text{if } h_{N,mode}(m,n) = 1; \\ f_\Omega(k) & \text{otherwise.} \end{cases}$$

2. Otherwise, if $k = 0$, the trivial pattern $H_0$ is used, namely $\widetilde{P}_{proc} = \widetilde{P}$.

3. The residual samples are computed in the frequency domain using $\widetilde{P}_{proc}$, quantised (to obtain $\widetilde{C}$), dequantised and inverse transformed (to obtain $\widetilde{X}_{rec}$). $\widetilde{C}$ and $\widetilde{X}_{rec}$ are used to compute the RD cost relative to the current element defined by $k$. A temporary solution is considered as $k^o$ such that $\widetilde{P}_{proc}$ returns the current minimum RD cost.

4. If $k < K$, the index $k$ is incremented by 1 and the algorithm goes back to step 1. Otherwise the element at minimum RD cost is output, identified by its optimal index $k^o$.

Only the optimal index $k^o$ needs to be encoded in the bitstream when using this algorithm, to be extracted at the decoder side and used to select the optimal solution. Note that this is the only overhead required by the entire proposed method in this subsection.

## 5.3 Results

The approaches presented in this chapter were implemented in the context of the HEVC reference software version HM 12.0 [85]. All tests were performed under high quality conditions, namely using four low QP values of 2, 5, 7 and 12. Tests are performed both on still image coding (the Kodak test set was used) and on video sequences. In the second case, sequences at different resolutions were tested.

Two sets of results were obtained and are presented in this section. In the first set, Algorithm 1 presented in subsection 5.2.1 is tested, referred to as Pattern Selection in the tables. This algorithm consists in selecting the best masking pattern for a given block in an RD sense, under the assumption that all discarded coefficients as an effect of using a certain pattern are set to zero. The index to identify the optimal pattern is signalled in the bitstream.

Results for still image coding are reported in Table 5-E. Following the convention used in the rest of this thesis, negative values of the BD-rates correspond to a more efficient encoding than the anchor; this can be an effect of achieving higher coding qualities, lower bitrates or both.

As the approach only affects intra-prediction, the all-intra configuration was used when testing video sequences. Results of such tests are reported in Table 5-F.

Up to $-3.7\%$ BD-rate gains were obtained when coding still images, with average $-2.9\%$ gains. The method outperforms conventional HEVC in all experiments. Similarly when coding video sequences, gains were reported in all tests, with up to $-4.40\%$ BD-rate gains.

The second set of results presents performance of Algorithm 2 in subsection 5.2.2 making use of context-aware look-up tables. This method was also tested both on still images and video sequences. In the latter case a broad variety of sequences was used, including more resolutions of standard sequences, and also less conventional types of

Table 5-E: Results for Still Image Coding (Pattern Selection)

| **Image** | **BD-rates** |
|---|---|
| Stone building | −2.0% |
| Red door | −2.5% |
| Hats | −3.7% |
| Girl in red | −2.7% |
| Motocross bikes | −2.0% |
| Sailboat | −2.4% |
| Window | −3.1% |
| Market place | −1.8% |
| Spinnakers | −2.8% |
| Sailboat race | −2.8% |
| Pier | −2.2% |
| Couple on beach | −2.7% |
| Mountain stream | −2.0% |
| Water rafters | −2.2% |
| Girl | −2.9% |
| Tropical key | −2.8% |
| Monument | −2.8% |
| Model in black | −2.2% |
| Lighthouse | −2.4% |
| Mustang | −2.4% |
| Portland headlight | −2.5% |
| Barn and pond | −2.3% |
| Parrots | −3.6% |
| Chalet | −2.2% |

Table 5-F: Results for Video Coding in All-Intra Configuration (Pattern Selection)

| **Resolution** | **Sequence** | **BD-rates** |
|---|---|---|
| $832 \times 480$ | Mobisode2 | −4.4% |
| | Keiba | −2.7% |
| | Partyscene | −1.0% |
| $416 \times 240$ | Basketballpass | −2.7% |
| | Racehorses | −2.6% |

content specifically interesting for high quality conditions. In particular, two screen content signals were tested, namely sequences containing computer generated scenery such as graphic overlays, large amounts of texts, scrolling subtitles and so on. High quality coding is particularly relevant for this kind of content, for instance in the case of screen mirroring applications or in medical imaging. Eventually the approach was also tested on two UHD sequences at a resolution of $3840 \times 2160$ luma samples. High quality video coding is relevant in this case mostly due to the increasing demand of the general public for content at increasingly high resolutions at very high levels of quality.

Full results are presented in Table 5-G (for still image coding) and Table 5-H (for video coding). In the case of video sequences, compared with the results in Table 5-F obtained with the Pattern Selection approach, the tests in Table 5-H report an increase

Table 5-G: Results for Still Image Coding (Look-up Tables)

| **Image** | **BD-rates** |
|---|---|
| Stone building | −2.4% |
| Red door | −2.9% |
| Hats | −3.5% |
| Girl in red | −2.8% |
| Motocross bikes | −2.0% |
| Sailboat | −2.7% |
| Window | −3.2% |
| Market place | −2.3% |
| Spinnakers | −3.3% |
| Sailboat race | −3.1% |
| Pier | −2.6% |
| Couple on beach | −3.1% |
| Mountain stream | −2.2% |
| Water rafters | −2.4% |
| Girl | −3.2% |
| Tropical key | −3.1% |
| Monument | −3.0% |
| Model in black | −2.4% |
| Lighthouse | −2.8% |
| Mustang | −2.9% |
| Portland headlight | −2.8% |
| Barn and pond | −2.6% |
| Parrots | −3.6% |
| Chalet | −2.4% |

Table 5-H: Results for Video Coding in All-Intra Configuration (Look-up Tables)

| **Resolution** | **Sequence** | **BD-rates** |
|---|---|---|
| Standard sequences | | |
| 2560 × 1600 | Nebuta | −1.5% |
| | Steamlocomotive | −2.3% |
| 1920 × 1080 | Basketballdrive | −4.6% |
| | Kimono | −1.5% |
| | BQTerrace | −4.3% |
| 1280 × 720 | Johnny | −1.7% |
| | FourPeople | −1.5% |
| | KristenAndSara | −1.7% |
| 832 × 480 | Mobisode2 | −5.2% |
| | Keiba | −2.7% |
| | Partyscene | −2.3% |
| 416 × 240 | Basketballpass | −3.2% |
| | Racehorses | −2.4% |
| Screen content | | |
| 1024 × 768 | ChinaSpeed | −1.4% |
| | SlideShow | −1.2% |
| Ultra high-definition content | | |
| 3840 × 2160 | Lupo Boa | −2.0% |
| | Veggie Fruits | −1.5% |

in performance efficiency in 4 out of 5 of the tests. In one case (Partyscene) the BD-rate gains are more than doubled when using context-aware look-up tables with respect to coefficient masking and discarding. In general the method is capable of obtaining consistent gains.

It is also interesting to report some observations on the percentages of TUs that were encoded using the modified encoder scheme with respect to the total. In fact, the proposed algorithm involves an RD decision to choose whether a given TU should make use of transform domain prediction processing or not. Consider for instance the Hats image: in average 71% of all TUs were encoded using a masking pattern and a corresponding context-aware look-up table. Similar percentages were obtained in the other images in the test set.

Finally, RD curves obtained by the approach for the Basketball pass and Keiba sequences are shown in the plots in Figure 5.6. Note that while some gains are obtained at all tested values of the QP, best performances are obtained for the highest qualities (Qp equal to 2) corresponding to the right-most points in the plots in the figure. A gain of up to 5.7 dB is obtained for the average Y PSNR of the Basketballpass sequence in this test point.

Finally, some further experiments were performed on this approach to evaluate its impact when testing other coding configurations. In fact, while the method directly affects only intra-predicted blocks (and as such has the greatest impact when testing the all-intra configuration), its effects have an impact also when using the low-delay or random access configurations. Even though when using such configurations most of the blocks are predicted using motion compensation, improving intra-prediction has an effects on the coding efficiency due to two factors: (i) a better reference frame is available when computing the inter-prediction, and correspondingly more efficient coding can be expected; (ii) intra-prediction modes are also tested in P and B slices which means. At this purpose, some results for these configurations are reported in Table 5-I for test sequences at a resolution of $1920 \times 1080$. From this table it is clear that the effects of the approach are beneficial to the performance of the encoder also when using such configurations.

Table 5-I: Video Coding in Random Access and Low Delay Configurations.

| Random access | | |
|---|---|---|
| **Resolution** | **Sequence** | **BD-rates** |
| 1920 × 1080 | Basketballdrive | −4.1% |
| | Kimono | −0.7% |
| | BQTerrace | −3.1% |
| Low delay | | |
| **Resolution** | **Sequence** | **BD-rates** |
| 1920 × 1080 | Basketballdrive | −4.3% |
| | Kimono | −0.6% |
| | BQTerrace | −3.2% |

## 5.4 Conclusions

An analysis of intra-prediction methods for video compression under high quality conditions was presented in this chapter. The study was based on two modified encoder and decoder schemes, in which original and prediction blocks are directly transformed and the residuals are computed in the transform domain. Such schemes allow a detailed study

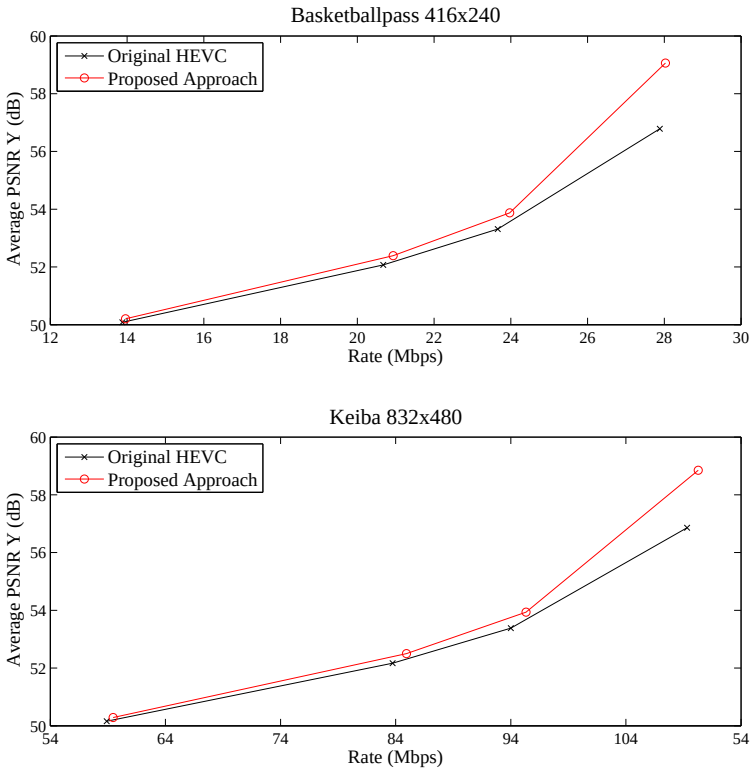



Figure 5.6: PSNR vs Bitrate curve for the Basketballpass and Keiba sequences.

of the performance of each intra-prediction method directly in the transform domain. The state-of-the-art HEVC standard was used as a base for the implementation. The analysis showed that high frequency components are typically difficult to predict using conventional intra-prediction methods, often resulting in high bitrates of the encoded signal. The similarity between prediction and original coefficients at particular frequency components was studied, resulting in very low similarity values in many cases especially at high frequencies.

A novel approach was proposed to improve the efficiency of high quality video coding based on such an analysis. An additional stage of transform domain processing was introduced during encoding and decoding before the residual computation. The processing is based on coefficient masking: selected patterns are used at this purpose to classify each frequency component in the transformed prediction blocks. Some components are preserved whereas other components are discarded and possibly replaced with more informative content.

At this purpose two methods were proposed and detailed in the chapter based on this approach. In the first method the encoder computes the optimal pattern in an RD sense for each block, and then discards all masked coefficients. An index is transmitted to signal the pattern to the decoder side. Conversely a second method was proposed based on context-aware look-up tables. The coefficients that are masked out by applying a certain pattern are replaced by means of context-aware look up tables. The derivation of such tables and related algorithm was illustrated in the chapter. The methods were shown capable of consistently improving compression efficiency with respect to conventional HEVC both when coding still images and video sequences.

# Chapter 6

# Conclusions and Future Developments

The consumer and professional technology markets have evolved in the past decades at an incredible pace. New products and services have been introduced which allow capturing, delivery or consumption of multimedia content under a variety of conditions that were unimaginable only ten years ago. Real-time streaming of high definition content over the internet, distribution of media at ultra high-definition (UHD) resolutions, screenshot sharing or screen mirroring among a variety of devices, high quality video recording and instant sharing: these are only some of a growing number of new applications that are directly affected by video coding.

Following from this observation, the fundamental aim of the research illustrated in this thesis is that of providing faster and more efficient video coding schemes under the broadest possible conditions. In the introduction of this thesis a number of factors were briefly illustrated as a list of the major obstacles that make meeting this goal so difficult. In this thesis, several contributions were proposed which attempt at approaching such a goal from different perspectives, while also trying to address (at least partially) all of these factors.

In particular:

1. Low-complexity techniques were proposed, which allow for efficient video coding even using very limited resources. This is typical of many consumer applications which can benefit from higher compression efficiencies at lower computational costs.

2. Most video coding algorithms target the encoding of natural scenery (namely camera captured signals) at "conventional" resolutions, under medium-to-low quality constraints. A broader set of conditions was targeted during the development of some of the contributions presented in this thesis. A variety of contents and different applications (either in the professional and consumer market) were considered.

3. Higher compression efficiency was investigated; several novel methods were proposed in the thesis with the general goal of providing higher qualities of the decoded sequence and/or lower bitrates of the compressed data, to allow for more efficient content storage and distribution.

4. The activities of the main bodies operative in video coding standardisation were followed during the entire course of the research described in this thesis. All contributions were investigated, developed and tested taking into account such activities. Some of the research illustrated here also helped the development and validation of next generation video coding standards.

Finally, a brief summary of the major contributions presented in the thesis is reported here. These are listed according to the chapter in which they are presented as follows:

1. In Chapter 3, methods to achieve faster encoding of video signals were investigated, to target more efficient compression at lower costs in terms of resources and computational power. In particular, methods in the context of adaptive precision motion estimation were proposed. The approaches are based on a geometrical characterisation of the residual error surface computed after integer-precision motion estimation. An original metric was defined to characterise the curvature of this

surface based only on the actual information at integer-valued locations available to the encoder. It was shown that an accurate classification of the blocks can be performed based on this metric, by taking into account the actual impact of fractional refinements on the prediction accuracy of each block. Several algorithms were proposed to perform binary classification of optimal prediction accuracy based on this concept. Among such methods, a novel approach was proposed where the classifier is trained on-line using a number of blocks and then progressively updated to adapt to local conditions of the sequence. The method was shown being capable of adapting to local conditions, limiting the performance losses while at the same time maximising the savings in terms of computational time.

2. In Chapter 4, a new module referred to as Enhanced Inter-prediction (EIP) was proposed based on the idea that inter-predicted samples can be further processed on-the-fly to improve prediction accuracy by means of parametric matrix functions. The module was investigated to target different goals and under different conditions. In particular, the method was investigated with the goal of improving compression efficiency under medium-to-low levels of quality. A particular case of the approach was proposed referred to as Shifting Transformation. An algorithm to derive the optimal parameter and corresponding distortion gain for each pair of original and prediction blocks was defined, and the corresponding method was studied and integrated in the context of rate-distortion optimisation as it is used in the AVC standard. Considerable gains in compression efficiency were obtained and are reported in the thesis. Finally the EIP was also considered in the context of the next generation HEVC standard. The module was shown capable of improving the coding efficiency of this new standard under particular conditions and following some adaptation. Different implementations of the EIP were proposed to influence specific components of the coding scheme.

3. In Chapter 5, techniques for improving video coding efficiency under high quality constraints were studied and proposed. An analysis of conventional intra-prediction

techniques as used in HEVC was presented based on the similarity between prediction and original signals at different frequency components. At this purpose, modified encoder and decoder schemes were proposed in which the residual samples are directly computed in the transform domain. The analysis showed that conventional techniques suffer from poor prediction of high frequency components and also highlighted a correlation between the performance of such methods and local characteristics of the coded blocks. Methods to improve the coding efficiency were studied based on this analysis. An additional stage of transform domain processing was introduced during encoding and decoding. In particular it was shown that coefficient masking can be successfully used on the transformed intra-predicted signals to remove components that are badly correlated with the original signal. A framework to perform the selection of such components based on masking patterns was proposed. Finally, it was proved that synthetic content can be injected in the transformed prediction signal to increase redundancy and eventually improve the performance of the encoder. Context-aware look-up tables were shown to be a valuable tool to produce this synthetic content. The methods were tested on a variety of different content, including UHD sequences and screen content. Consistent gains were reported when using the method compared with existing techniques.

All of the contributions presented in this thesis can be improved and are subject of further developments.

In the context of Chapter 3, the proposed curvature metric can be modified to better adapt to particular fast integer-precision motion estimation algorithms by taking into account the different characteristics of these algorithms. Also the binary classifier may be extended to a multiclass classifier capable to discriminate among different fractional precision levels. Finally, more efficient decision functions may be defined and used for the classification.

In the context of Chapter 4, different matrix functions may be investigated as an alternative to the Shifting Transformation, to improve the outcomes of the EIP depending

on the particular application. The module may be also improved by considering and embedding the prediction of the EIP parameters in the coding scheme, similar to the motion vector prediction used when encoding the motion information. Finally, techniques to hide the transmission of such parameters within other portions of the bitstream may be studied and formulated.

In the context of Chapter 5, the selection of frequency components that are preserved or discarded may be more efficiently designed in order to adapt to local characteristics of the signal. This might happen for instance by means of on-line training of the pattern selection algorithm, or a more sophisticated coefficient classification not necessarily based on patterns may also be formulated. Eventually, other methods to derive the synthetic content that is injected in the signal can also be investigated.

# List of Publications

**Academic Papers**

S. G. Blasi, M. Mrak and E. Izquierdo, "Frequency Domain Intra-Prediction Analysis and Processing for High Quality Video Coding", *IEEE Transactions on Circuits and Systems for Video Technologies*, 2014 (considered for publication subject to minor revision)

S. G. Blasi, I. Zupancic and E. Izquierdo, "Fast Motion Estimation Discarding Low-Impact Fractional Blocks", *European Signal Processing Conference (EUSIPCO)*, 2014 (accepted for presentation)

S. G. Blasi, M. Mrak and E. Izquierdo, "Masking of Transformed Intra-predicted Blocks for High Quality Image and Video Coding", *IEEE International Conference on Image Processing (ICIP)*, 2014 (accepted for presentation)

M. Naccari, S. G. Blasi, M. Mrak and E. Izquierdo, "Improving Inter Prediction in HEVC with Residual DPCM for Lossless Screen Content Coding", *Picture Coding Symposium (PCS)*, San Jose (California, USA) 2013

S. G. Blasi, E. Peixoto and E. Izquierdo, "Mode Decision with Enhanced Inter-Prediction in HEVC", *IEEE International Conference on Image Processing (ICIP)*, Melbourne (Australia), 2013

S. G. Blasi, E. Peixoto and E. Izquierdo, "Enhanced Inter-Prediction using Merge Prediction Transformation in the HEVC Codec", *IEEE International Conference on Acoustics,*

*Speech and Signal Processing (ICASSP)*, Vancouver (Canada), 2013

S. G. Blasi, E. Peixoto and E. Izquierdo, "Enhanced Inter Prediction via Shifting Transformation in the H.264/AVC", *IEEE Transactions on Circuits and Systems for Video Technologies*, Vol. 24, N. 4, pp. 735–740, 2012

S. G. Blasi and E. Izquierdo, "Residual error curvature estimation and adaptive classification for selective sub-pel precision motion estimation", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto (Japan), 2012

**Contributions to Standardisation**

M. Naccari, M. Mrak, A. Gabriellini, S. G. Blasi and E. Izquierdo, "Inter-Prediction Residual DPCM", Joint Collaborative Team on Video Coding (JCT-VC), Doc. JCTVC-M0442, Incheon (Korea), 2013

**Patents**

S. G. Blasi, E. Peixoto and E. Izquierdo, "A Method of Coding a Video Signal", U.K. Patent GB2500023, 2013

# References


[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[2] I. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia.* Wiley, 2003.

[3] J. D. Gibson, T. Berger, T. Lookabaugh, R. Baker, and D. Lindbergh, *Digital Compression for Multimedia: Principles and Standards.* Morgan Kaufmann Publishers, 2006.

[4] ITU-T, "ITU-T Recommendation BT.709, Parameter Values for the HDTV Standards for Production and International Programme Exchange," ITU-T, Tech. Rep., 2012.

[5] D. H. Hubel, *Eye, Brain and Vision.* Scientific American Library, 1988.

[6] C. J. van den Branden Lambrecht, *Vision Models and Applications to Image and Video Processing.* Kluwer Academic Publishers, 2001.

[7] P. Read and M. Meyer, *Restoration of Motion Picture Film. Conservation and Museology.* Butterworth-Heinemann, 2000.

[8] M. Rolfs, D. Jonikaitis, H. Deubel, and P. Cavanagh, "Predictive Remapping of Attention Across Eye Movements," *Nature Neuroscience*, vol. 14, no. 2, pp. 252–256, 2011.

[9] K. Noland, "The Human Visual System and High Frame Rate Television," in *EBU UHDTV - Voices and Choices*, 2013.

[10] ITU-T, "ITU-T Recommendation BT.2020, Parameter Values for Ultra-High Definition Television Systems for Production and International Programme Exchange,"

ITU-T, Tech. Rep., 2012.

[11] ITU-T, "ITU-T Recommendation H.261, Video Codec for Audiovisual Services at p×64 kbit/s," ITU-T, Tech. Rep., 1993.

[12] EBU, "High Definition (HD) Image Formats for Television Production," EBU, Tech. Rep., 2004.

[13] Z. Wang and A. C. Bovik, "Mean squared error: Love It or Leave It? A New Look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: from Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[15] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale Structural Similarity for Image Quality Assessment," in *Conference on Signals, Systems and Computers*, vol. 2, 2003, pp. 1398–1402 Vol.2.

[16] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "Study of Subjective and Objective Quality Assessment of Video," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1427–1441, 2010.

[17] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in Image/Video Quality Assessment," *Electronics Letters*, vol. 44, no. 13, pp. 800–801, 2008.

[18] G. J. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[19] G. Bjontegaard, "Improvements of the BD-PSNR model," ITU-T SG16/Q6, 35th VCEG Meeting, Doc.VCEG-AI11, 2008.

[20] G. Bjontegaard, "Coding Improvement by Using $4 \times 4$ Blocks for Motion Vectors and Transform," ITU-T, Tech. Rep., 1997.

[21] C. Dai, *Selected Topics in Video Coding and Computer Vision.* ProQuest, 2008.

[22] G. K. Wallace, "The JPEG Still Picture Compression Standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[23] D.-Y. Kim, K.-H. Han, and Y.-L. Lee, "Adaptive Single-Multiple Prediction for H.264/AVC Intra Coding," *IEEE Transactions on Circuits and Systems for Video*

*Technology*, vol. 20, no. 4, pp. 610–615, 2010.

[24] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, "Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 378–401, 2005.

[25] A.-C. Tsai, A. Paul, J.-C. Wang, and J.-F. Wang, "Intensity Gradient Technique for Efficient Intra-Prediction in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 5, pp. 694–698, 2008.

[26] J. Jain and A. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Transactions on Communications*, vol. 29, no. 12, pp. 1799 – 1808, 1981.

[27] T.-S. Choi and J.-N. Kim, "Real-time Video Coding," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 2, 1999.

[28] W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation," *IEEE Transactions on Image Processing*, vol. 4, no. 1, pp. 105 –107, 1995.

[29] H.-S. Wang and R. Mersereau, "Fast Algorithms for the Estimation of Motion Vectors," *IEEE Transactions on Image Processing*, vol. 8, no. 3, pp. 435 – 438, 1999.

[30] S. M. Jung, S. C. Shin, H. Baik, and M. S. Park, "Efficient Multilevel Successive Elimination Algorithms for Block Matching Motion Estimation," *IEEE Proceedings on Vision, Image and Signal Processing*, vol. 149, no. 2, pp. 73 – 84, 2002.

[31] C. Zhu, W.-S. Qi, and W. Ser, "Predictive Fine Granularity Successive Elimination for Fast Optimal Block-matching Motion Estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 213 –221, 2005.

[32] C.-H. Lee and L.-H. Chen, "A Fast Motion Estimation Algorithm based on the Block Sum Pyramid," *IEEE Transactions on Image Processing*, vol. 6, no. 11, pp. 1587 –1591, 1997.

[33] F. Kossentini, Y.-W. Lee, M. J. T. Smith, and R. K. Ward, "Predictive RD Optimized Motion Estimation for Very Low Bit-rate Video Coding," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 9, pp. 1752–1763, 1997.

[34] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing Images using the Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850 –863, 1993.

[35] R. Li, B. Zeng, and M. L. Liou, "A New Three-step Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438 – 442, 1994.

[36] S. Zhu and K.-K. Ma, "A New Diamond Search Algorithm for Fast Block-matching Motion Estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287 –290, 2000.

[37] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based Search Pattern for Fast Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 349 –355, 2002.

[38] A. M. Tourapis, O. C. Au, and M. L. Liou, "Highly Efficient Predictive Zonal Algorithms for Fast Block-matching Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 934 – 947, 2002.

[39] B. Girod, "Motion-compensating Prediction with Fractional-pel Accuracy," *IEEE Transactions on Communications*, vol. 41, no. 4, pp. 604 –612, 1993.

[40] K.-H. Lee, J.-H. Choi, B.-K. Lee, and D.-G. Kim, "Fast Two-step Half-pixel Accuracy Motion Vector Prediction," *Electronics Letters*, vol. 36, no. 7, pp. 625 –627, 2000.

[41] K. Panusopone and D. M. Baylon, "An Analysis and Efficient Implementation of Half-pel Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 8, pp. 724 – 729, 2002.

[42] J. W. Suh and J. Jeong, "Fast Sub-pixel Motion Estimation Techniques Having Lower Computational Complexity," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 3, pp. 968 – 973, 2004.

[43] J. W. Suh, J. Cho, and J. Jeong, "Model-based Quarter-pixel Motion Estimation with Low Computational Complexity," *Electronics Letters*, vol. 45, no. 12, pp. 618 –620, 2009.

[44] J. Cho, J. W. Suh, G. Jeon, and J. Jeong, "Selective Mathematical Modelling

Algorithm for Quarter-pixel Motion Estimation," *Electronics Letters*, vol. 48, no. 9, pp. 491 –493, 2012.

[45] R. Krishnamurthy, J. W. Woods, and P. Moulin, "Frame Interpolation and Bidirectional Prediction of Video Using Compactly Encoded Optical-Flow Fields and Label Fields," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 5, pp. 713 – 726, 1999.

[46] M. Haller, A. Krutz, and T. Sikora, "Robust Global Motion Estimation using Motion Vectors of Variable Size Blocks and Automatic Motion Model Selection," in *IEEE International Conference on Image Processing*, 2010, pp. 737–740.

[47] M. K. Gullu, "Block Motion Estimation using Luminance Transform Based Matching," in *IEEE Signal Processing, Communication and Applications Conference*, 2008, pp. 1–4.

[48] Y. Zhou, X. Sun, H. Bao, and S. Li, "Weighted Motion Estimation for Efficiently Coding Scene Transition Video," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 2004, pp. iii–361–4 vol.3.

[49] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.

[50] K. R. Rao and P. C. Yip, *The Transform and Data Compression Handbook*. CRC Press, 2000.

[51] V. Britanak, P. C. Yip, and K. R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic Press, 2010.

[52] C. Yeo, Y. H. Tan, and Z. Li, "Low-complexity Mode-dependent KLT for Block-based Intra Coding," in *IEEE International Conference on Image Processing*, 2011, pp. 3685–3688.

[53] A. Saxena and F. C. Fernandes, "DCT/DST-Based Transform Coding for Intra Prediction in Image/Video Coding," *IEEE Transactions on Image Processing*, vol. 22, no. 10, pp. 3974–3981, 2013.

[54] C. Zhang and D. Florencio, "Analyzing the Optimality of Predictive Transform Coding Using Graph-Based Models," *IEEE Signal Processing Letters*, vol. 20, no. 1,

pp. 106–109, 2013.

[55] ITU-T, "ITU-T Recommendation H.120, Codecs for Videoconferencing using Primary Digital Group Transmission," ITU-T, Tech. Rep., 1993.

[56] MPEG, "ISO/IEC 11172, Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s," MPEG, Tech. Rep., 1993.

[57] MPEG, "ISO/IEC 13818 - Generic Coding of Moving Pictures and Associated Audio Information," MPEG, Tech. Rep., 1995.

[58] ITU-T, "ITU-T Recommendation H.263, Video Coding for Low Bitrate Communication," ITU-T, Tech. Rep., 1996.

[59] S. Srinivasan, P. Hsu, T. Holcomb, K. Mukerjee, S. L. Regunathan, B. Lin, J. Liang, M.-C. Lee, and J. Ribas-Corbera, "Windows Media Video 9: overview and applications," *Signal Processing: Image Communication*, vol. 19, no. 9, pp. 851 – 875, 2004.

[60] H. Kalva and J. B. Lee, "The VC-1 Video Coding Standard," *IEEE Transactions on MultiMedia*, vol. 14, no. 4, pp. 88–91, 2007.

[61] BBC, "Dirac Specification," BBC, Tech. Rep., 2008.

[62] SMPTE, "ST 2042, VC-2 Video Compression," SMPTE, Tech. Rep., 2008.

[63] Google, "VP8 Data Format and Decoding Guide," Google, Tech. Rep., 2011.

[64] Google. VP9 Video Codec. [Online]. Available: http://www.webmproject.org/vp9/

[65] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, "The Latest Open-source Video Codec VP9 - An Overview and Preliminary Results," in *Picture Coding Symposium*, 2013, pp. 390–393.

[66] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders," in *Picture Coding Symposium, 2013*, 2013, pp. 394–397.

[67] ITUT, "Itut recommendation h.264, advanced video coding for generic audiovisual services," ITUT, Tech. Rep., 2003.

[68] DVB, "Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system," DVB, Tech. Rep., 2012.

[69] DVB, "Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications," DVB, Tech. Rep., 2013.

[70] Blu-Ray Disc Association, "Audio visual application format specifications for bd-rom," Blu-Ray Disc Association, Tech. Rep., 2011.

[71] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, 2003.

[72] ITU. JM Reference Software. [Online]. Available: http://http://iphome.hhi.de/suehring/tml/

[73] I. Richardson, *The H.264 Advanced Video Compression Standard*. Wiley, 2010.

[74] I. Richardson, "H.264/MPEG-4 Part 10: Intra Prediction White Paper," Tech. Rep., 2002.

[75] T. Shiodera, A. Tanizawa, and T. Chujoh, "Block Based Extra/Inter-Polating Prediction for Intra Coding," in *IEEE International Conference on Image Processing*, vol. 6, 2007, pp. VI – 445–VI – 448.

[76] H. Lakshman, B. Bross, H. Schwarz, and T. Wiegand, "Fractional-sample Motion Compensation using Generalized Interpolation," in *Picture Coding Symposium*, 2010, pp. 530–533.

[77] J. M. Boyce, "Weighted Prediction in the H.264/MPEG AVC Video Coding Standard," in *International Symposium on Circuits and Systems*, vol. 3, 2004, pp. III–789–92 Vol.3.

[78] A. Tanizawa and T. Chujoh, "Efficient Weighted Prediction Parameter Signaling using Parameter Prediction and Direct Derivation," in *IEEE International Conference on Image Processing*, 2013, pp. 1900–1903.

[79] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity Transform and Quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 598–603, 2003.

[80] R. G. de Oliveira and R. L. De Queiroz, "Intra Prediction versus Wavelets and Lapped Transforms in an H.264/AVC coder," in *IEEE International Conference*

*on Image Processing*, 2008, pp. 137–140.

[81] M. Wang, K. Ngan, and L. Xu, "Efficient H.264/AVC Video Coding with Adaptive Transforms," *IEEE Transactions on Multimedia*, vol. PP, no. 99, pp. 1–1, 2014.

[82] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive Deblocking Filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 614–619, 2003.

[83] ITU-T VCEG, "Joint Call for Proposals on Video Compression Technology," VCEG-AM91 and MPEG N11113, 2010.

[84] ITU-T, "Itu-t recommendation h.264, advanced video coding for generic audiovisual services," ITU-T, Tech. Rep., 2003.

[85] ITU. HM Reference Software. [Online]. Available: https://hevc.hhi.fraunhofer.de/HM-doc/

[86] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[87] J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards-Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, 2012.

[88] K. Ugur and J. Lainema, "Updated results on HEVC still picture coding performance," document n. JCTVC-M0041, Incheon (South Korea), 2012.

[89] R. Weerakkody and M. Mrak, "High Efficiency Video Coding for Ultra High Definition Television," in *NEM Summit*, vol. 1, 2013, pp. 9–14.

[90] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra Coding of the HEVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1792–1801, 2012.

[91] Y. Zheng, M. Coban, and M. Karczewicz, "Simplified Intra Smoothing," document n. JCTYC-C234, Guangzhou (China), 2010.

[92] K. Ugur, A. Alshin, E. Alshina, F. Bossen, W.-J. Han, J. H. Park, and J. Lainema, "Interpolation Filter Design in HEVC and its Coding Efficiency - Complexity Anal-

ysis," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1704–1708.

[93] P. Helle, S. Oudin, B. Bross, D. Marpe, M. O. Bici, K. Ugur, J. Jung, G. Clare, and T. Wiegand, "Block Merging for Quadtree-Based Partitioning in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1720–1731, 2012.

[94] J.-L. Lin, Y.-W. Chen, Y.-P. Tsai, Y.-W. Huang, and S. Lei, "Motion Vector Coding Techniques for HEVC," in *IEEE International Workshop on Multimedia Signal Processing*, 2011, pp. 1–6.

[95] W.-J. Han, J. Min, I.-K. Kim, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y. M. Hong, M.-S. Cheon, N. Shlyakhov, K. McCann, T. Davies, and J.-H. Park, "Improved Video Compression Efficiency Through Flexible Unit Representation and Corresponding Extension of Coding Tools," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1709–1720, 2010.

[96] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand, "Transform Coding Techniques in HEVC," *IEEE Journal of Selected Topics in Signal Processing*, vol. PP, no. 99, pp. 1–1, 2013.

[97] A. Fuldseth, G. Bjntegaard, M. Budagavi, and V. Sze, "CE10: Core Transform Design for HEVC," document n. JCTVC-G495, Geneva (Switzerland), 2011.

[98] K. Ugur and B. O., "Performance Evaulation of DST in Intra-Prediction," document n. JCTVC-I0582, Geneva (Switzerland), 2012.

[99] C.-M. Fu, C.-Y. Chen, Y.-W. Huang, and S. Lei, "Sample Adaptive Offset for HEVC," in *IEEE International Workshop on Multimedia Signal Processing*, 2011, pp. 1–5.

[100] W.-S. Kim and D.-K. Kwon, "Improved Sample Adaptive Offset for HEVC," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1700–1703.

[101] H. Lakshman, C. Rudat, M. Albrecht, H. Schwarz, D. Marpe, and T. Wiegand, "Conditional Motion Vector Refinement for Improved Prediction," in *Picture Coding Symposium*, 2012, pp. 497–500.

[102] J. Ribas-Corbera and D. L. Neuhoff, "Optimizing Motion-vector Accuracy inBlock-based Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 497–511, 2001.

[103] J. Cho, G. Jeon, J. Suh, and J. Jeong, "Sub-Pixel Motion Estimation Scheme Using Selective Interpolation," *IEICE Transactions on Communications*, vol. 91, pp. 4078–4080, 2008.

[104] J. W. Suh, J. Cho, and J. Jeong, "Model-based Quarter-pixel Motion Estimation with Low Computational Complexity," *Electronics Letters*, vol. 45, no. 12, pp. 618–620, 2009.

[105] Q. Zhang, Y. Dai, and C. C. J. Kuo, "Direct Techniques for Optimal Sub-Pixel Motion Accuracy Estimation and Position Prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1735–1744, 2010.

[106] J. J. Koenderink and A. J. van Doorn, "Surface shape and curvature scales," *Image and vision computing*, vol. 10, no. 8, pp. 557–564, 1992.

[107] K. R. Rao, D. N. Kim, and J. J. Hwang, *Video Coding Standards*. Springer, 2014.

[108] H. Li and R. Forchheimer, "A Transformed Block-based Motion Compensation Technique," *IEEE Transactions on Communications*, vol. 43, no. 234, pp. 1673–1676, 1995.

[109] M. Narroschke and R. Swoboda, "Extending HEVC by an Affine Motion Model," in *Picture Coding Symposium*, 2013, pp. 321–324.

[110] T. Lin, K. Zhou, X. Chen, and S. Wang, "Arbitrary Shape Matching for Screen Content Coding," in *Picture Coding Symposium, 2013*, 2013, pp. 369–372.

[111] N. J. Higham, *Functions of Matrices - Theory and Computation*. SIAM, 2008.

[112] D. Marpe, H. Schwarz, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Suhring, M. Winken, and T. Wiegand, "Video Compression Using Nested Quadtree Structures, Leaf Merging, and Improved Techniques for Motion Representation and Entropy Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1676–1687, 2010.

[113] Y. Zheng, M. Coban, and M. Karczewicz, "Common Test Conditions and Software

Reference Configurations," document n. JCTVC-H1100, San Jose (USA), 2012.

[114] K. Sharman, N. Saunders, and J. Gamei, "CE1: Test 1 - Rectangular transform units for 4:2:2 (and AHG7 benchmarks)," document n. JCTVC-L0182, Geneva (Switzerland), 2013.

[115] D. Flynn, J. Sole, and T. Suzuki, "High Efficiency Video Coding (HEVC) Range Extensions text specification: Draft 2," document n. JCTVC-L1005, Geneva (Switzerland), 2013.

[116] E. Wige, G. Yammine, P. Amon, A. Hutter, and A. Kaup, "Sample-based Weighted Prediction with Directional Template Matching for HEVC lossless coding," in *Picture Coding Symposium*, 2013, pp. 305–308.

[117] Y. H. Tan, C. Yeo, and Z. Li, "Residual DPCM for Lossless Coding in HEVC," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 2021–2025.

[118] M. Zhou, W. Gao, M. Jiang, and H. Yu, "HEVC Lossless Coding and Improvements," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1839–1843, 2012.

[119] P. Amon, A. Hutter, E. Wige, and A. Kaup, "RCE2: Sample-based Weighted Intra Prediction for Lossless Coding," document n. JCTVC-M0052, Incheon (South Korea), 2013.

[120] E. Wige, G. Yammine, P. Amon, A. Hutter, and A. Kaup, "Pixel-based Averaging Predictor for HEVC Lossless Coding," in *IEEE International Conference on Image Processing*, 2013, pp. 1806–1810.

[121] MPEG, "ISO/IEC N13828 - Draft Requirements for Future Extensions of HEVC in Coding Screen Content and Medical Visual Content," MPEG, Tech. Rep., 2013.

[122] MPEG, "ISO/IEC N13829 - Draft Call for Proposals for Coding of Screen Content and Medical Visual Content," MPEG, Tech. Rep., 2013.

[123] J. Han, V. Melkote, and K. Rose, "Transform-domain Temporal Prediction in Video Coding: Exploiting Correlation Variation across Coefficients," in *IEEE International Conference on Image Processing*, 2010, pp. 953–956.

[124] J. Han, V. Melkote, and K. Rose, "Transform-domain Temporal Prediction in

Video Coding with Spatially Adaptive Spectral Correlations," in *IEEE 13th International Workshop on Multimedia Signal Processing*, 2011, pp. 1–6.

[125] Kodak. Kodak Test Image set. [Online]. Available: http://r0k.us/graphics/kodak/

[126] S.-H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307, 2007.

[127] M. Kokare, B. N. Chatterji, and P. Biswas, "Comparison of Similarity Metrics for Texture Image Retrieval," in *Conference on Convergent Technologies for the Asia-Pacific Region*, vol. 2, 2003, pp. 571–575 Vol.2.

[128] T.-C. Chen, Y.-W. Huang, C.-Y. Tsai, B.-Y. Hsieh, and L.-G. Chen, "Architecture Design of Context-Based Adaptive Variable-Length Coding for H.264/AVC," *IEEE Transactions on Circuits and Systems*, vol. 53, no. 9, pp. 832–836, 2006.

[129] I. Richardson, "H.264/AVC Context Adaptive Variable Length Coding White Paper," Tech. Rep., 2002.

[130] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, 2003.

[131] Y.-L. Lee, K.-H. Han, and G. Sullivan, "Improved Lossless Intra Coding for H.264 / MPEG4 AVC," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2610–2615, 2006.

[132] Y. H. Tan, C. Yeo, and Z. Li, "Residual DPCM for Lossless Coding in HEVC," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 2021–2025.

[133] M. Zhou, W. Gao, M. Jiang, and H. Yu, "Hevc lossless coding and improvements," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1839–1843, 2012.

[134] W. Gao, M. Zhou, P. Amon, and S. Lee, "HEVC Range Extensions Core Experiment (RCE) 2: Intra Prediction for Lossless Coding," document n. JCTVC-L1122,

Geneva (Switzerland), 2013.

# Appendix A

# Entropy Coding in AVC and HEVC

This appendix presents a brief introduction to context adaptive variable length coding (CAVLC) and context-adaptive binary arithmetic coding (CABAC), focusing on their implementation in the AVC and HEVC standards.

## A.1 Context Adaptive Variable Length Coding

While alternative entropy coding schemes may be supported in certain versions of AVC, Context Adaptive Variable Length Coding (CAVLC) is supported by any AVC decoder and it is widely used mostly due to its relatively low computational complexity [128]. In the context of this thesis, CAVLC was used in all cases when working with AVC. CAVLC is based on the use of variable length coding (VLC) tables, namely look-up tables that associate the possible values of the parameter being encoded to codewords of a variable number of bits. The idea of CAVLC is that of allowing a multitude of possible VLC tables to encode each parameter. The choice of which look-up table to use is based on local characteristics of the array such as previously encoded arrays or previously encoded

parameters within the same array.

When using CAVLC, coefficients are encoded in groups of 16 following the same partitioning in $4 \times 4$ blocks used during transform and quantisation. First, they are ordered in arrays following the zig-zag scan illustrated in Figure 1.1, and then they are input to the entropy coder.

In particular the process of encoding an array of coefficients starts by counting the number of non-zero values and trailing ones [129]. The trailing ones are the number of $+1$ and/or $-1$ in the array (regardless of their sign). These are counted in reverse order starting from the end of the array and up to a maximum of 3. In case there are more than 3 trailing ones, only the first 3 are initially encoded while the remaining are considered along with the rest of the non-zero coefficients in the array. The numbers nT and nL of non-zero coefficients in the neighbouring blocks on the left and top of the current block respectively are also considered, and a parameter nC is computed as the average of nT and nL. If one of these blocks is not available nC is set to nT or nL (whichever is available), if both are not available nC is set to 0. Finally a VLC table is extracted depending on nC, which is used to encode collectively the number of non-zero coefficients and trailing ones in the block. For each trailing one, a bit is also encoded to signal its sign, in reverse order starting from the end of the array.

The non-zero coefficients (excluding the trailing ones) are then coded,in reverse order

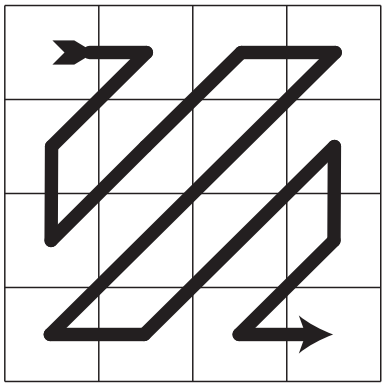


Figure 1.1: Zig-zag scan.

starting from the end of the array. In case less than 3 trailing ones were previously signaled, the first non-zero coefficient cannot be equal to $\pm 1$ (or it would have been counted among the trailing ones); for this reason in this case the magnitude of the first non-zero coefficient is decreased by one, to reduce the number of bits needed to encode it. One out of seven possible VLC tables may be used to encode the non-zero coefficients, referred to as VLC-$x$, $x = 0, ..., 6$. Small values of $x$ correspond to tables more efficient for coding small coefficients, while higher values of $x$ correspond to tables that code more efficiently large coefficients. The first non-zero coefficient is coded using either VLC-0 or VLC-1 depending on the number of non-zero coefficients and trailing ones. The remaining non-zero coefficients are coded, and $x$ is updated accordingly to the magnitude of the last-encoded coefficient, to adapt the choice of the VLC table to the local characteristics of the array.

Then the total number of zero coefficients preceding the last non-zero coefficient is encoded using an appropriate VLC table. Finally, the number of neighbouring zero coefficients preceding each non-zero coefficient (including the trailing ones) is encoded, again in reverse order starting from the end of the array.

All other parameters except the coefficients are encoded in AVC using Exp-Golomb codes. Exp-Golomb coding mode assigns each possible value of the currently encoded variable $v$ to a fixed code of regular construction, following two successive steps. In the first step, the actual value $v$ of the parameter is mapped to an unsigned integer $z$. This mapping can happen in three possible ways depending on the parameter being coded. In case the parameter can only assume positive integer values (for instance, the intra-prediction mode from 0 to 3 in an Intra-16 $\times$ 16 macroblock), most of the times $z = v$. Conversely in case the parameter can assume signed integer values (for instance, the motion vector difference components), most of the times it is mapped as:

$$z = 2|v|$$

| z | Codeword |
|---|---|
| 0 | 1 |
| 1 | 010 |
| 2 | 011 |
| 3 | 00100 |
| 4 | 00101 |
| 5 | 00110 |
| 6 | 00111 |

if $v < 0$, or:

$$z = 2|v| - 1,$$

otherwise. In all other cases, a specific mapping is defined to assign each possible value of $v$ to values of $z$, starting from the most probable value of $v$ mapped to $z = 0$ and incrementing $z$ for more probable values of $v$.

In the second step $z$ is binarised and encoded in the bitstream. The Exp-Golomb codewords used for the binarisation progress in logical order as:

$$[0000...]\,[1]\,[CODE]\,.$$

where the codeword starts with an initial run of $N$ zeros, where $N = \lfloor \log_2(z+1) \rfloor$. This is followed by a 1. Finally a binary string of exactly $N$ bits is inserted of value $CODE = z + 1 - 2^M$. In total, each codeword is formed of $2N + 1$ bits. The first codewords for $z$ from 0 to 6 are shown in Table A.1.

## A.2 Context Adaptive Binary Arithmetic Coding

Context Adaptive Binary Arithmetic Coding (CABAC) was already implemented in some versions of AVC as an entropy coder alternative to CAVLC, and it is the entropy coder of choice in HEVC. Variable length coding suffers from well-known problems due to the fact that it always assigns an integer number of bits to each possible value of a symbol.

For instance, assuming that a binary index $i$ is being encoded where $p(i = 0) = 0.95$ and $p(i = 1) = 0.05$, even if the information content of $i = 0$ is as low as $-log_2(0.95) = 0.075$, still variable length coding would assign at least 1 bit to represent this value. Arithmetic coding solves this issue by encoding a sequence of symbols into a single fractional number that represents the entire message.

The initial idea of arithmetic coding was already proposed by Shannon in the late 40s and was later developed to its modern form in the late 80s. In its simplest form, arithmetic coding works by iteratively assigning intervals to symbols within a range of real numbers delimited by two extremes $a$ and $b$. One non-overlapping interval $[a_i b_i)$ is assigned to each symbol according to its probability of occurrence in such a way that the union of all intervals form the entire range or: $[a_0 b_0) \cup [a_1 b_1) \cup ...[a_N b_N) = [ab)$. Initially the range $a = 0$ and $b = 1$ is used, and at each encoded symbol $i$ the range shortens to the interval $a = a_i$, $b = b_i$. After all symbols are coded the entire sequence is represented by any fractional number within the obtained range of values.

The arithmetic coder included in HEVC is binary in the sense that variables are binarised before being coded, and each binary value (bin) is coded independently, which means only two ranges are assigned at each step. It is context adaptive in the sense that the probabilities used to define the ranges during the encoding are not fixed, but they can be selected after encoding each symbol, from a set of available probability models based on statistics of previously coded symbols.

First the currently encoded variable is binarised to obtained a codeword. Binarisation may happen following several methods depending on the variable. For example Exp-Golomb codes similar to the ones used for CAVLC in AVC can be used, or fixed-length codes, or other techniques. Each bin in the codeword is input to the arithmetic coder independently. Binarisation has two main advantages compared to applying arithmetic coding to the original set of variables and their possible values. More obviously, binary arithmetic coding is much less computationally expensive than its $n$-nary counterpart. Especially in case of probability models that require the coder to keep track of several

past values of a variable, the resulting entropy coder can be extremely demanding in terms of resources and complexity. But more importantly, by considering each bin in the codeword independently, the coder can perform context modeling at a bin-by-bin level. Each bin in a codeword can be assigned its probability and make use of specific models. This is extensively used in the CABAC algorithm in HEVC. In most cases when more bins result from the binarisation of a variable, only the most frequently observed bin (i.e. the first bit in the codeword) is coded using conditional probabilities, while fixed zero-order probability models are used for other less frequently observed bins.

A context model is then selected for each bin. Each context model is assigned a unique context index that is formed of a fixed component (which identifies the type of variable that is currently being encoded) plus a variable offset (that identifies the local characterisation of the currently encoded bin). While more details on this can be found in the literature [130], for the purpose of this thesis it is interesting to notice that the variable offset is derived for most variables by means of a simple selection that only takes into account a few past values of the bin. In most cases only two past values are used as illustrated in the following example. Assume that the encoder is coding the flag to signal whether a CU is split or not. The context index is first initialised to a fixed value $\gamma_{SplitFlag}$ depending on the sole condition that the current bin is derived from a CU split flag. A variable offset between 0 and 2 is then computed as $\alpha(SplitFlag_{Top}, SplitFlag_{Left})$, namely depending on the values of the split flags $SplitFlag_{Top}$ and $SplitFlag_{Top}$ in neighbouring blocks on top and on the left of the current CU respectively. Finally the total context index to code the current bin is obtained as $\chi = \gamma_{SplitFlag} + \alpha(SplitFlag_{Top}, SplitFlag_{Left})$.

For each value of the context index $\chi$, a probability state is stored and used. To allow fast computations, limit the amount of needed resources and at the same time obtain accurate probability estimates, the probability state of the least probable symbol (LPS) is represented by means of a fixed number of allowed probability values between 0 and 0.5. 64 allowed probability state values are used in the CABAC implementation in HEVC,

denoted in reverse order as $\sigma_i$ with $i = 0, ..., 63$ where $\sigma_0$ approximately correspond to a probability of 0.5 and increasing values of $\sigma_i$ correspond to lower values of the probability towards 0. Considering also an additional bit required to signal whether the current LPS is a 0 or a 1, in total 7 bits are needed to represent the information required for each value of $\chi$.

The probability states $\sigma_i$ for each context model are first initialised before encoding based on some prior knowledge on each bin and source statistics, and also depending on the slice type and the QP used for quantisation. This is because the slice type and the value of the QP both have a strong impact on the actual content of the bitstream and as such it makes sense to allow some appropriate pre-adaptation of the initial probability states. Then after encoding each bin using the context model identified by $\chi$, the probability state is updated accordingly. If the bin value was equal to the MPS, the probability is decremented to the previous allowed value, or $\sigma_{i-1}$. Conversely if a bin equal to the current LPS was coded, the probabilty is incremented to a value greater than $\sigma_i$. Appropriate transition rules are defined at this purpose. Finally in case probability $\sigma_0$ is reached at a point during the encoding (approximately corresponding to a probability of 0.5), in case another bin equal to the LPS value is encoded, the LPS value is changed (from 0 to 1 or vice versa).

# Appendix B

# Residual DPCM for Lossless Screen Content Coding

This appendix briefly presents some methods to address the coding of screen content under lossless constraints.

## B.1 Background

Screen content refers collectively to image or video sequences which contain artificial (computer generated) content, either exclusively or when it is mixed with more conventional camera captured content. Examples of screen content are screenshots from computer applications, slideshow presentations, videos containing subtitles or other superimposed objects, and so on. Conventional video coding methods relying on quantisation of high frequency components are not optimal due to the fact that the sharp edges and high contrast areas typical of this type of content may result degraded. As a result the intelligibility of text or other small details may be compromised. For these reasons lossless coding techniques are typically preferable for these applications.

Lossless video coding may be achieved in a conventional video coding scheme by simply skipping the transform and quantisation steps. The encoder only performs intra or inter-prediction and the corresponding residual samples are directly input to the entropy coder. Support for lossless video coding by means of transform and quantisation bypass is already supported by the HEVC standard. On the other hand, due to the strong characterisation of screen content, it makes sense to investigate techniques specifically for the purpose of increasing the efficiency of screen content compression under these conditions.

Differential pulse code modulation (DPCM) has been previously used to better exploit spatial redundancy in the signal. Sample-by-sample residual DPCM of intra-predicted residuals was proposed [131] in the context of AVC lossless coding. When using this technique, instead of performing conventional intra-prediction each residual sample is predicted from neighbouring residual samples in the vertical or horizontal direction, in case the intra-prediction is performed with an angular mode following one of these two directions. Average bitrate reductions of $-12\%$ were reported using this technique compared with conventional AVC lossless coding. The method was later extended and adapted to the HEVC standard [132] achieving on average $-6.5\%$ gains.

Other methods have been proposed to increase the efficiency of lossless video coding. Recently, sample based angular intra-prediction (SAP) [133] was introduced for HEVC lossless coding. When using this technique each sample inside a PU is predicted using samples in the same PU. The prediction samples can be extracted from the PU at different angles not limited to the horizontal or vertical prediction directions. By performing intra-prediction on a sample-by-sample basis within a PU, spatial redundancy can be better exploited which results in up to $-11.8\%$ bitrate reductions compared to conventional HEVC lossless coding.

## B.2 Inter Residual DPCM

When performing inter-prediction it is assumed that all samples within the current block move approximately along the same trajectory. This assumption leads to poor prediction accuracy in particular along sharp edges. In screen content it is reasonable to expect that inter-predicted residuals still present some correlation along image edges. Such correlation may be further exploited to achieve higher compression efficiency.

In particular consider a block R of $M \times N$ residual samples as they are output from inter-prediction, and refer to each element in R as $r(i, j)$. Two DPCM modes are defined referred to as horizontal and vertical mode respectively. When using vertical inter residual DPCM, all samples in the block R are predicted from neighbouring samples immediately above in the same column. Samples in the first row are left unchanged. Formally:

$$\hat{r}_{ver}(i,j) = \begin{cases} r(i,j) & i = 0 \\ r(i,j) - r(i-1,j) & \text{otherwise} \end{cases}$$

A similar expression is used to define horizontal inter residual DPCM: samples in the first column in the block are left unchanged, while all other samples are predicted from the sample immediately on the left in the same row.

The proposed residual DPCM mode can be integrated in a conventional HEVC scheme as follows. The encoder computes the optimal inter-prediction for each PU depending on the inter-prediction mode. The residual block R is computed independently for each PU as the difference between original and prediction blocks. Then, both proposed inter residual DPCM modes are applied, resulting in the processed residual blocks $\mathrm{R}_{ver}$ and $\mathrm{R}_{hor}$. The prediction distortions for the original residual samples and the processed samples obtained using either of the two proposed residual DPCM modes

are then computed and compared to select the solution at minimum distortion. Note that the encoder is also allowed the option of not using residual DPCM, to efficiently encode residual blocks that are uncorrelated, namely where no further spatial prediction is needed. In particular, the sum of absolute values of all elements in R, $\mathrm{R}_{ver}$ or $\mathrm{R}_{hor}$ is calculated and used as distortion metric. Finally, an index to signal the solution at minimum distortion is coded in the bitstream at a PU level.

At the decoder side this index is extracted to apply the correct residual DPCM mode (if used) on each block of residual samples. Two inverse residual DPCM modes can be defined to obtain the original residuals from $\mathrm{R}_{ver}$ and $\mathrm{R}_{hor}$ respectively. In particular, inverse vertical inter residual DPCM can be defined as:

$$r(i,j) = \sum_{k=0}^{i-1} \hat{r}_{ver}(k,j).$$

Inverse horizontal residual DPCM can be defined equivalently where the summation is performed across each row of samples.

A problem with this approach as defined here is that when applying the inverse residual DPCM modes, samples in the $i$-th row (for the vertical mode) or $j$-th column (for the horizontal mode) depend on the previous $i-1$ or $j-1$ samples in the same row or column. Especially for large block sizes this leads to an increase in computational complexity, and also generates dependencies among samples that are very distant among each other. Such effects may not be acceptable in some applications. In order to prevent these issues, the prediction by means of DPCM can be limited to group of samples of a maximum predefined size $L$. Instead of performing prediction on the whole block, the vertical and horizontal modes are adapted to perform a refresh after each consecutive

group of $L$ samples as follows. Vertical residual DPCM is modified as:

$$\hat{r}_{ver}(i,j) = \begin{cases} r(i,j) & i = 0, L, 2L, ... \\ r(i,j) - r(i-1,j) & \text{otherwise} \end{cases}$$

and correspondingly at the decoder side the residuals can be reconstructed as:

$$r(i,j) = \sum_{k=\lfloor i/L \rfloor L}^{i-1} \hat{r}_{ver}(k,j).$$

Equivalent expressions for the horizontal residual DPCM mode can be easily derived when using a refresh with a size $L$.

## B.3 Results

In lossless mode, no distortion between reconstructed and original signal results from the encoding and decoding processes due to the fact that the decoded sequence is mathematically identical to the original. As a consequence, bitrate reduction in percentage (denoted as $\Delta$ bitrate in this section) with respect to conventional HEVC is used as the only performance metric under these conditions (as opposite to BD-rates, used in the case of lossy coding). Negative values of $\Delta$ bitrate corresponds to an improvement with respect to the benchmark. The approach was implemented in the context of the HEVC HM reference software version HM 10-RExt-2.0 [85]. Screen content sequences considered within the HEVC test set were used for the experiments, referred to as Class F and Class SC [134]. All tests were performed using the random access configuration.

Results of the methods are presented in Table 2-A. Results on the left side of the table correspond to the case when no refresh is considered, namely when $L$ is set to the whole height or width of the PU (when using vertical or horizontal modes respectively). Conversely on the right side of the table, performance of the method when refresh is

Table 2-A: Results of Inter Residual DPCM.

| | **No Refresh** | **Refresh** ($L = 8$) |
|---|---|---|
| | **Δ Bitrate (%)** | **Δ Bitrate (%)** |
| Class F | −5.1 | −4.3 |
| Class SC | −3.7 | −3.5 |
| Average | −4.4 | −3.9 |

used with $L = 8$ are presented.

By using residual DPCM, average bitrate reductions of −4.4% may be obtained as reported in the table. When using refresh of the prediction with a parameter $L = 8$, a decrease in performance of the approach is expected, due to the fact that some samples are not predicted by DPCM but are transmitted without further processing. Nonetheless the refresh allows less dependencies among samples at spatially distant locations.